

magazine
msdn®

DATABASE PROGRAMMING

The 'Juneau' Database Project

Jamie Laflen and Barclay Hill 32

New Features in the Entity Framework June CTP

Srikanth Mandadi 46

Build Great Experiences on Any Device with OData

Shayne Burgess 54

PLUS:

No Browser Left Behind: An HTML5

Adoption Strategy

Brandon Satrom 62

Build MVVM Applications in F#

Chris Marinos 68

Visual Studio ALM Rangers—Reflections on Virtual Teams

Brian Blackman and Willy-Peter Schaub 76

COLUMNS

CUTTING EDGE

Software Disasters: Recovery and
Prevention Strategies

Dino Esposito page 6

WINDOWS WITH C++

The Thread Pool Environment

Kenny Kerr page 12

DATA POINTS

Second-Level Caching in the Entity
Framework and AppFabric

Julie Lerman page 16

FORECAST: CLOUDY

Reporting on Diagnostics Data

Joseph Fultz page 24

THE WORKING PROGRAMMER

Multiparadigmatic .NET, Part 10:

Choosing an Approach

Ted Neward page 84

UI FRONTIERS

Touch for Text

Charles Petzold page 92

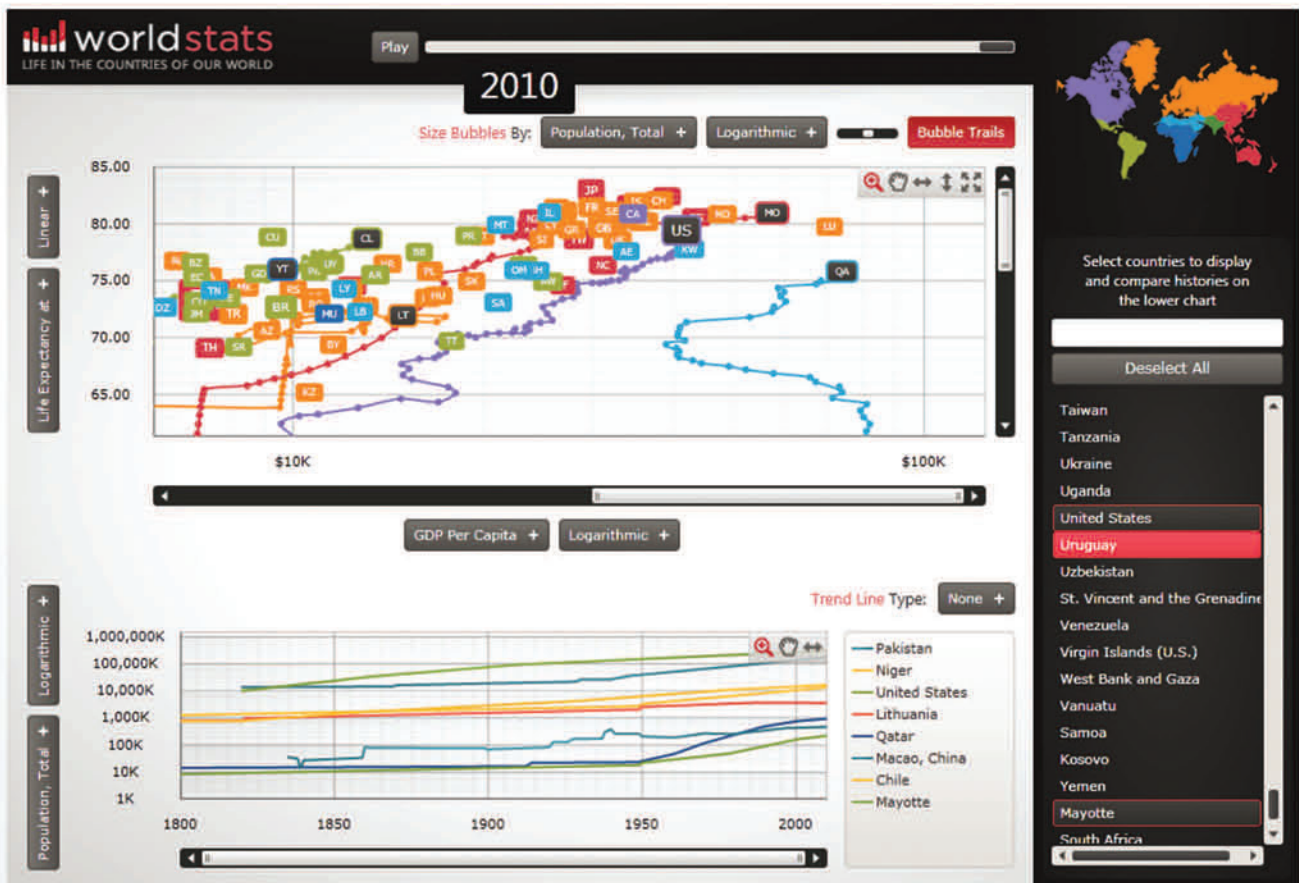
DON'T GET ME STARTED

Development Is Design

David Platt page 96

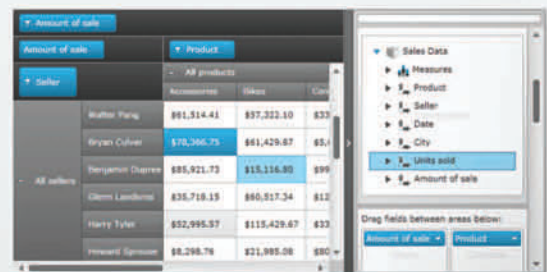
NetAdvantage® ULTIMATE

REPORTING, DATA VISUALIZATION AND LOB UI CONTROLS FOR ASP.NET, WINDOWS FORMS, JQUERY/HTML5, WPF, SILVERLIGHT AND WINDOWS PHONE 7



INFRAGISTICS MOTION FRAMEWORK™

Delivering a great user experience in Windows Presentation Foundation (WPF) and Microsoft Silverlight business intelligence applications requires more than styling, it requires giving your application's end users greater insight into the story of their data.



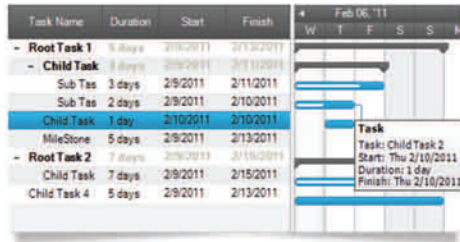
OLAP PIVOT GRID DATA VISUALIZATION

Work with multidimensional data from your OLAP cubes, data warehouses and Microsoft® SQL Server® Analysis Services.



ASP.NET GAUGE

Whether it's for a sidebar gadget or an internal portal such as SharePoint®, gauges play a crucial role on any dashboard.



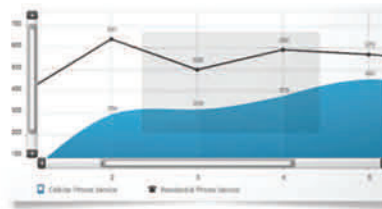
WINDOWS FORMS GANTT CHARTS

Deliver a Microsoft Project-style user experience to your users, with the composite tabular/timeline view of your scheduling data.

Product ID	Product Name	Product Number
▼ Equals	▼ Contains	▼ Contains
Clear Filter		
Equals	Adjustable Race	AR-5381
Does not equal	Bearing Ball	BA-8327
Greater than	3B Ball Bearing	BE-2349
Less than	Headset Ball Bearings	BE-2908
Greater than or equal to	Blade	BL-2036
Less than or equal to	LL Crankarm	CA-5965
318	ML Crankarm	CA-6738
319	HL Crankarm	CA-7457
320	Chainring Bolts	CB-2903
321	Chainring Nut	CN-6137

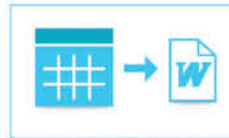
JQUERY

The most robust and forward-thinking product we have based on emerging Web technologies including HTML5 and CSS 3.



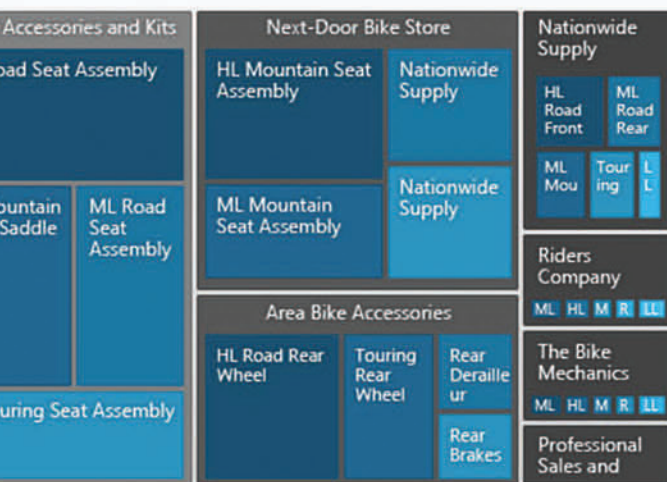
CHARTING

Make business charting quick and easy with fast, interactive, and vivid visuals across every .NET platform.



EXPORT TO MICROSOFT® WORD

New class library can create Word documents and stream xamGrid™ contents from Silverlight to a Word document.



SILVERLIGHT DATA VISUALIZATION

Use the Silverlight Data Visualization treemap control to communicate differences in data points with different pattern identifications.



WINDOWS PHONE 7

Visually and functionally designed to build eye-catching, high-end user experiences that take your mobile applications to the next level on the Microsoft® Windows Phone® 7.



SCAN HERE for an exclusive look at Ultimate!
www.infragistics.com/ult

TAKE YOUR APPLICATIONS TO THE NEXT LEVEL WITH OUR TOOLS
INFRAGISTICS.COM/ULTIMATE

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE



dtSearch®

Instantly Search Terabytes of Text

"Bottom line: dtSearch manages a terabyte of text in a single index and returns results in less than a second"

InfoWorld

"Covers all data sources ... powerful Web-based engines"

eWEEK

"Lightning fast ... performance was unmatched by any other product"

Redmond Magazine

For hundreds more reviews and developer case studies, see www.dtSearch.com

Highlights hits in a wide range of data, using dtSearch's own file parsers and converters

- Supports MS Office through 2010 (Word, Excel, PowerPoint, Access), OpenOffice, ZIP, HTML, XML/XSL, PDF and more
- Supports Exchange, Outlook, Thunderbird and other popular email types, including nested and ZIP attachments
- Spider supports static and dynamic web data like ASP.NET, MS SharePoint, CMS, PHP, etc.
- API for SQL-type data, including BLOB data

25+ full-text & fielded data search options

- Federated searching
- Special forensics search options
- Advanced data classification objects

APIs for C++, Java and .NET through 4.x

- Native 64-bit and 32-bit Win / Linux APIs; .NET Spider API
- Content extraction only licenses available

Desktop with Spider

Web with Spider

Network with Spider

Engine for Win & .NET

Publish (portable media)

Engine for Linux

Ask about fully-functional evaluations!

The Smart Choice for Text Retrieval® since 1991

www.dtSearch.com • 1-800-IT-FINDS



msdn®

magazine

SEPTEMBER 2011 VOLUME 26 NUMBER 9

LUCINDA ROWLEY Director

KIT GEORGE Editorial Director/mmeditor@microsoft.com

KERI GRASSL Site Manager

MICHAEL DESMOND Editor in Chief/mmeditor@microsoft.com

DAVID RAMEL Technical Editor

SHARON TERDEMAN Features Editor

WENDY GONCHAR Managing Editor

KATRINA CARRASCO Associate Managing Editor

SCOTT SHULTZ Creative Director

JOSHUA GOULD Art Director

CONTRIBUTING EDITORS Dino Esposito, Joseph Fultz, Kenny Kerr, Julie Lerman, Dr. James McCaffrey, Ted Neward, Charles Petzold, David S. Platt

RedmondMediaGroup

Henry Allain President, Redmond Media Group

Matt Morollo Vice President, Publishing

Doug Barney Vice President, Editorial Director

Michele Imgrund Director, Marketing

Tracy Cook Online Marketing Director

ADVERTISING SALES: 508-532-1418/mmorollo@1105media.com

Matt Morollo VP Publishing

Chris Kourtoglou Regional Sales Manager

William Smith National Accounts Director

Danna Vedder Microsoft Account Manager

Jenny Hernandez-Asandas Director Print Production

Serena Barnes Production Coordinator/msdnadproduction@1105media.com

1105 MEDIA

Neal Vitale President & Chief Executive Officer

Richard Vitale Senior Vice President & Chief Financial Officer

Michael J. Valenti Executive Vice President

Abraham M. Langer Senior Vice President, Audience Development & Digital Media

Christopher M. Coates Vice President, Finance & Administration

Erik A. Lindgren Vice President, Information Technology & Application Development

Carmel McDonagh Vice President, Attendee Marketing

David F. Myers Vice President, Event Operations

Jeffrey S. Klein Chairman of the Board

MSDN Magazine (ISSN 1528-4859) is published monthly by 1105 Media, Inc., 9201 Oakdale Avenue, Ste. 101, Chatsworth, CA 91311. Periodicals postage paid at Chatsworth, CA 91311-9998, and at additional mailing offices. Annual subscription rates payable in US funds are: U.S. \$35.00, International \$60.00. Annual digital subscription rates payable in U.S. funds are: U.S. \$25.00, International \$25.00. Single copies/back issues: U.S. \$10, all others \$12. Send orders with payment to: *MSDN Magazine*, PO, Box 3167, Carol Stream, IL 60132, email MSDNmag@1105service.com or call (847) 763-9560. **POSTMASTER:** Send address changes to *MSDN Magazine*, PO, Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or XPO Returns: PO, Box 201, Richmond Hill, ON L4B 4R5, Canada.

Printed in the U.S.A. Reproductions in whole or part prohibited except by written permission. Mail requests to "Permissions Editor," c/o *MSDN Magazine*, 4 Venture, Suite 150, Irvine, CA 92618.

Legal Disclaimer: The information in this magazine has not undergone any formal testing by 1105 Media, Inc. and is distributed without any warranty expressed or implied. Implementation or use of any information contained herein is the reader's sole responsibility. While the information has been reviewed for accuracy, there is no guarantee that the same or similar results may be achieved in all environments. Technical inaccuracies may result from printing errors and/or new developments in the industry.

Corporate Address: 1105 Media, Inc., 9201 Oakdale Ave., Ste 101, Chatsworth, CA 91311, www.1105media.com

Media Kits: Direct your Media Kit requests to Matt Morollo, VP Publishing, 508-532-1418 (phone), 508-875-6622 (fax), mmorollo@1105media.com

Reprints: For single article reprints (in minimum quantities of 250-500), e-prints, plaques and posters contact: PARS International, Phone: 212-221-9595, E-mail: 1105reprints@parsintl.com, www.magreprints.com/QuickQuote.asp

List Rental: This publication's subscriber list, as well as other lists from 1105 Media, Inc., is available for rental. For more information, please contact our list manager, Merit Direct. Phone: 914-368-1000; E-mail: 1105media@meritdirect.com; Web: www.meritdirect.com/1105

All customer service inquiries should be sent to MSDNmag@1105service.com or call 847-763-9560.

Microsoft



Printed in the USA

techxtend.com
866-719-1528



programmer's
paradise



Embarcadero RAD Studio XE2

The ultimate application development suite for Windows, Mac, mobile and Web

by Embarcadero

Embarcadero® RAD Studio XE2 Professional is designed for software developers and teams building PC, Mac, kiosk and mobile applications with or without embedded and local database persistence. RAD Studio includes Delphi®, C++Builder®, and RadPHP™ providing you with everything needed for fast native Windows, Mac OS X, .NET, PHP, Web and iOS development.

NEW
VERSION!

Professional Ed.
Paradise #
CGI 15501A01

CALL

techxtend.com/embarcadero

UltraEdit

The world's #1 text editing solution is also the world's most affordable!

by IDM Computer Solutions

UltraEdit is the world's standard in text editors. Millions use UltraEdit as the ideal text/hex/programmers editor on any platform — Windows, Mac, or Linux!

Features include syntax highlighting for nearly any programming language; powerful Find, Replace, Find in Files, and Replace in Files; FTP support, sort, column mode, hex, macros/scripting, large file handling (4+ GB), projects, templates, Unicode, and more.



Named User
1-24 Users
Paradise #
184 01201A01

\$59.95

techxtend.com/idm

VMware vSphere 5 Essentials Kit Bundle

VMware vSphere is the industry-leading virtualization platform for building cloud infrastructures that enables users to run business critical applications with confidence and respond faster to their business.

vSphere accelerates the shift to cloud computing for existing datacenters, while also underpinning compatible public cloud offerings paving the way for the only hybrid cloud model. With over 250,000 customers worldwide and the support of over 2500 applications from more than 1400 ISV partners, VMware vSphere is the trusted platform for any application.



NEW
VERSION
5!

CALL

techxtend.com/vSphere



Spread 5 for Windows Forms by GrapeCity PowerTools

- World's best selling .NET Spreadsheet
- Import/export native Microsoft Excel files with full formatting
- Extremely flexible printing and export options including PDF
- Extensible formula support, including Microsoft Excel functions
- Hundreds of chart styles for enhanced data visualization
- Powerful user interface and flexible data connectivity
- WYSIWYG spreadsheet designers, quick-start wizard and chart designers
- Royalty-free licensing

Upgrade
Paradise #
F02 01101A01

\$936.99

techxtend.com/grapecity

Kingston DataTraveler G3 16 GB Flash Drive

by Kingston Technology

The new generation of a Kingston best-seller is here! With capacities up to 32GB, the reliable DataTraveler Generation 3 (G3) is ideal for your important documents, music, video clips and favorite photos that can be stored and retrieved in a flash.

Available in four fun colors by capacity, it's a perfect fit for the office, home, school and wherever you travel. A well-built cap protects the USB plug and your data. DataTraveler G3 also makes a great promotional item for your organization.



Paradise #
ZHI DQ1077

\$28.99

techxtend.com/kingston

TX Text Control 16.0

Word Processing Components

TX Text Control is royalty-free, robust and powerful word processing software in reusable component form.

- .NET WinForms and WPF rich text box for VB.NET and C#
- ActiveX for VB6, Delphi, VBScript/HTML, ASP
- File formats DOCX, DOC, RTF, HTML, XML, TXT
- PDF and PDF/A export, PDF text import
- Tables, headers & footers, text frames, bullets, structured numbered lists, multiple undo/redo, sections, merge fields, comments
- Ready-to-use toolbars and dialog boxes



Professional Edition
Paradise #
T79 12101A01

\$1,109.99

Download a demo today.

techxtend.com/textcontrol



Microsoft Visual Studio Professional 2010

by Microsoft

Microsoft Visual Studio 2010 Professional with MSDN Essentials Subscription is an integrated environment that simplifies creating, debugging and deploying applications. Unleash your creativity and bring your vision to life with powerful design surfaces and innovative collaboration methods for developers and designers. Work within a personalized environment, targeting a growing number of platforms, including Microsoft SharePoint and cloud applications and accelerate the coding process by using your existing skills.

Upgrade
Paradise #
M47 40201B02

\$483.99

techxtend.com/microsoft

Lenovo ThinkPad X220

by Lenovo

The ThinkPad X220 features the quality and performance Lenovo users have come to expect and increased audio, video and communications features that respond to the increased use of laptops as a multimedia and communications tools in business. The ultra-portable ThinkPad X220 comes equipped with a full-powered Intel processor with outstanding graphic performance. ThinkPad has improved on full-size keyboard design and nimble TrackPoint by adding an advanced buttonless touchpad.



Paradise #
ZHI GF0818

\$1,362.99

techxtend.com/lenovo

HP Laserjet P4014N Printer

by Hewlett Packard

Get reliable, fast, and affordable black-and-white printing for your workgroup! With print speeds of up to 45 pages per minute (ppm) and a 540 MHz processor, the HP LaserJet P4014 will help your workgroup operate at peak efficiency. Get consistent results and optimal performance with the printer's latest advancements in toner technology. Legendary HP reliability means worry-free printing that stands up to the demands of your office environment.



Paradise #
ZHI R83053

\$671.99

\$599.99

* Offer expires 9/30/11



Datawatch Monarch Professional 11.0

by Datawatch

No other desktop software tool provides Excel-based Business Intelligence (BI) and reporting as easily as Monarch.

Monarch lets anyone, regardless of technology skill level, acquire, customize and export data to Excel and other applications. Unlike traditional BI and reporting solutions, Monarch uses Report Mining to very easily acquire actionable data business users need, with no programming skills required.

Paradise #
D07 04201E01

\$952.99

techxtend.com/datawatch

Programmer's Paradise has a new name: TechXtend!



For nearly 30 years, Programmer's Paradise has served the software development and IT communities with the best selection of software, terrific values and a commitment to service excellence.

30 years... much has changed in that time!

However, one thing that won't ever change is that we will still be your "go to" source for software developer tools — AND you'll also be able to depend on TechXtend for all of your other IT needs.

Learn more about our new name: www.techxtend.com/techxtend

Win an iPad!

NO PURCHASE NECESSARY. Use offer code **WEBTRW09** when you place your order online or with your TechXtend/Programmer's Paradise representative and you'll automatically be entered into a drawing to win an iPad Wi-Fi 32GB.

For official rules, or to complete the online entry form: www.techxtend.com/tradewinds



Software, Systems, & Solutions — Since 1982

Prices subject to change. Not responsible for typographical errors.



'Mango': Innovation from the Inside Out

By now you may have noticed that Keith Ward is no longer gracing the Editor's Note page of *MSDN Magazine*. He has moved on to take the post of editor in chief of *Visual Studio Magazine*, where I know he's looking forward to doing more direct reporting and writing in the development space. I expect he'll also enjoy the editorial freedom that comes with running an independent publication.

I know all this because Keith and I have been colleagues for six or seven years now. I even worked for Keith briefly when he was editor of *Redmond* magazine. In that time, I've seen Keith launch new magazines, take principled stands, chase a lifelong dream and dive into covering not one, but two, entirely new fields. To say Keith is a tough act to follow is a testament to tough acts everywhere.

Keith's fingerprints are all over *MSDN Magazine*, from the talented lineup of regular columnists to the thoughtful coverage of complex technical subjects. The model he has adopted will no doubt continue to serve the magazine and its readers well going forward.

That said, the September issue of *MSDN Magazine* sits on the cusp of a strategic change of seasons in the Microsoft development space. Later this month, the long-awaited Microsoft BUILD Conference in Anaheim, Calif., promises to shed light on "Windows 8," HTML5 and JavaScript development, and the future direction of Silverlight and XAML. I'm not privy to confidential information, but watching the BUILD narrative unfold, I can't help but feel that Microsoft development could look very different a few years from now.

Maybe *MSDN Magazine* should look different, too. It's a conversation worth having, not just among the editors and publishers and stakeholders at Microsoft, but with the developers who rely on the magazine to keep them informed on the tools and technologies they work with every day. I hope to be able to have that conversation with you over the coming months and years, as we wade into what I have no doubt will be deep waters created by the September announcements at BUILD.

The Lessons of 'Mango'

The July launch of the Windows Phone 7 "Mango" update could be instructive as we look forward. As Blue Badge Insights founder and Microsoft Regional Director Andrew Brust noted in a July *Visual Studio Magazine* column, some of the most important innovation at Microsoft is coming not from rain-making business units and

products like Windows, Office and SharePoint, but from upstart teams like the Windows Phone group. And Microsoft is smartly adapting these innovations to other areas of the business. You need look no further than the Metro-inspired UI of Windows 8 to understand what Brust was talking about.

So when the Windows Phone 7 Mango release went live in July, there was some added import. The update is about more than winning market share from the Apple iOS and Google Android; it's about gauging what the Windows Phone group can achieve down the road. Yes, a poorly received update could blunt Windows Phone 7 adoption—but the greater harm might be to the innovation the group can bring to other areas of Microsoft.

Al Hilwa, program director for Application Development Software at research firm IDC, offered strong praise for the Mango update, which he said closed the gap between Windows Phone 7 and its competitors. "The key differentiator is the visual appeal of the platform," he said, calling the application updatable hubs on the home screen in Mango "one of my favorite features. I envision this to evolve into a scorecard of everything that's important at a glance, once applications are revised to take advantage of it."

It should be no surprise that Hilwa thought Microsoft did well on the developer front. "I think what the first year of Windows Phone shows us is that a strong developer ecosystem alone is not enough to achieve business success," he said. "It is, however, one of the hardest things to build, and Microsoft demonstrated that they're up to that part of the task."

There are challenges ahead. Microsoft must keep the basic programming model stable as it possibly faces a realignment with Windows 8, Hilwa said—and he added Microsoft would do better to produce a steady stream of feature improvements, rather than drop giant releases like Mango. Still, he predicted good things ahead for the platform.

"Given how late Microsoft started on this project, it has accomplished a great deal in less than a year," Hilwa noted. "Its chances will be helped significantly with a successful Windows 8 release in 2012, which will create synergies between the PC and the phone in new ways."

Visit us at msdn.microsoft.com/magazine. Questions, comments or suggestions for *MSDN Magazine*? Send them to the editor: mmeditor@microsoft.com.

© 2011 Microsoft Corporation. All rights reserved.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, you are not permitted to reproduce, store, or introduce into a retrieval system *MSDN Magazine* or any part of *MSDN Magazine*. If you have purchased or have otherwise properly acquired a copy of *MSDN Magazine* in paper format, you are permitted to physically transfer this paper copy in unmodified form. Otherwise, you are not permitted to transmit copies of *MSDN Magazine* (or any part of *MSDN Magazine*) in any form or by any means without the express written permission of Microsoft Corporation.

A listing of Microsoft Corporation trademarks can be found at microsoft.com/library/toolbar/3.0/trademarks/en-us.mspx. Other trademarks or trade names mentioned herein are the property of their respective owners.

MSDN Magazine is published by 1105 Media, Inc. 1105 Media, Inc. is an independent company not affiliated with Microsoft Corporation. Microsoft Corporation is solely responsible for the editorial contents of this magazine. The recommendations and technical guidelines in *MSDN Magazine* are based on specific environments and configurations. These recommendations or guidelines may not apply to dissimilar configurations. Microsoft Corporation does not make any representation or warranty, express or implied, with respect to any code or other information herein and disclaims any liability whatsoever for any use of such code or other information. *MSDN Magazine*, MSDN, and Microsoft logos are used by 1105 Media, Inc. under license from owner.

NEW!



OnTime11 is Here!

Ship Software OnTime

Project Management • Bug Tracking • Agile/Scrum

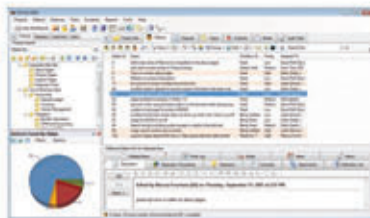
Get 2 Users Free... Forever! • Visit www.axosoft.com

Hosted



OnTime Now! Managing an Agile or Scrum dev team using a cloud-based solution has never been easier or more cost effective. Get started with 2 free users in just seconds at axosoft.com!

Windows



OnTime for Windows is installed in your own server environment. It uses a .NET & SQL back-end, integrates easily with Visual Studio, and is a total joy to use in the rich Windows client. Includes free SDK / APIs.

Web



OnTime Web provides you with an anywhere-anytime solution, whether you go with OnTime hosted or installed. This is what a web app is supposed to be - an intuitive UI, blazing fast reaction, and a powerful back end.

Free 2-user License • Free Download • Free Web Demo

 **axosoft.com**
800.653.0024



Software Disasters: Recovery and Prevention Strategies

As emphatic as it may sound, our lives depend on software. As users of various services ourselves, we know very well that software working as expected may really save our day. But software is getting more and more complex because it has to model the complexity of real-world processes. IT managers, architects and developers must cope with that.

In this article, I'll review practices that help fix a deteriorated system and show you patterns that may prevent a system from growing in an uncontrolled fashion. The term "big ball of mud" (BBM) was created years ago to refer to a system that's largely unstructured and padded with hidden dependencies between parts, with a lot of data and code duplication and an unclear identification of layers and concerns—a spaghetti code jungle. To read more about the BBM, I recommend an excellent piece of work by Brian Foote and Joseph Yoder from the University of Illinois at Urbana-Champaign, which you can download from bit.ly/nfPe2Y. In this paper, the authors don't condemn a BBM as the worst-ever problem; they just recommend that architects and developers be ready to face the BBM risk and learn how to keep it under control.

Software is getting more and more complex because it has to model the complexity of real-world processes.

The Dynamics of Software Systems

Some 30 years ago, in the beginning of the software era, applications were far easier to write than today. We had no need for GUIs. We didn't have to worry much about aesthetics, distribution needs, scalability issues or deployment concerns. And, last but certainly not least, we had much less business logic to implement.

At the time, software development was taken seriously and involved the best minds. Quite amazingly, in the early 1990s we already had most of the software development and architecture principles and patterns laid out. Arguably, very little has been "invented" since. Principles such as separation of concerns and dependency inversion, paradigms such as object-oriented programming (OOP) and aspect-orientation, and practices such as design for testability weren't developed in recent years. Many of them are being rediscovered and applied by today's architects and developers, but they have existed for decades.

But why have those principles and practices been lying in a sort of limbo for years? There may be many reasons for this, but a common developer refrain is, "It works, so why should I improve it (and pay the cost of wasting that extra time)?"

The Different Dynamics of the Java and Microsoft Worlds

In the mid-1990s, the advent of object-orientation and languages such as Java (more than C++) prompted many companies to restructure their software, moving larger and larger blocks of business logic out of databases and mainframes. Java developers, specifically in the enterprise space, had to deal with more complexity than that found in the rapid application development (RAD) typically associated with Visual Basic programming. It isn't surprising that aspect-orientation, dependency-injection frameworks (such as Spring) and object/relational mappers (such as Hibernate)—not to mention a long list of programming tools (such as NUnit, Ant, Javadoc, Maven and so on)—originated in the Java world. They were tools created by developers for developers to smooth the impact of complexity.

At the same time, in the Visual Basic world, RAD was all the rage. Visual Basic helped companies build nice front ends on top of stored procedures when not using mainframe code. The advent of Web services created an extra façade and gave mainframe code the much nicer name of "legacy services."

The OOP versus RAD debate over the past 15 years was a reasonable contention. When Microsoft introduced object-oriented features in Visual Basic, it was really hard to explain to developers why on earth those weird features were so important. And they actually weren't, given the relatively low level of complexity of those thin applications.

The .NET Revolution

The release of the Microsoft .NET Framework happened at a crucial time. It happened when some of the big companies that earlier opted for a RAD approach had seen enough of the Internet and had back-end systems old enough to justify a rebuild in light of new Internet-related business processes. At the same time, these companies found in the Microsoft platform a reliable, powerful and extensible framework. Of course, it took only a few years for .NET developers to be swamped with the same huge amount of complexity that Java colleagues experienced a decade earlier. Most .NET developers, though, grew with RAD principles and the RAD approach to development.

Blazing-Fast **GRID CONTROLS**

Optimized for .NET



Award-Winning Presentation Controls and Reporting Libraries



Learn more and download your **FREE** evaluation copy today
 Visit **DEVEXPRESS.COM/GRIDS**
 or Call Us (818) 844-3383

DevExpress™

PRESENTATION CONTROLS | REPORTING CONTROLS
 BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

All trademarks and registered trademarks are the property of their respective owners.

In a RAD world, you tend to prefer code that just works and build applications by adding blocks that are mostly self-contained. (And when they're not really self-contained, you make them that way by abusing code and data duplication).

The problem isn't with RAD as a programming paradigm. The real problem that's most likely to lead to the notorious BBM is applying RAD (or any other paradigm) without corrections that can keep the growth of the application, and subsequent multiplication of complexity, under control.

Clear Symptoms of a BBM

It seems to be a natural law of programming that every unit of software of any size (be it a class, a layer or an entire system) degrades. In their aforementioned paper, Foote and Yoder call it the "structural erosion" of software. Personally, I like to call it the biodegradability of software. As software units are maintained or changed, they become harder to further maintain or change. Regardless of the name, nowadays it's just part of the deal; you can't ignore it, and trying to do so will just cause damage.

Software biodegradability is tightly connected to piecemeal growth of projects. Incorporating a new requirement into an existing system—which was architected without that particular requirement—is *always* problematic. It doesn't mean that it can't, or shouldn't, be done—it just means that by adding a new requirement, you're changing the context. A single change doesn't usually have a dramatic impact on architecture, but when individual changes occur frequently, the context of the system changes over time and morphs into something that probably requires a different architecture. This process is also referred to as requirements churn.

Likewise, piecemeal growth is part of the deal, and not being ready to cope with that is one of the deadly sins of modern software architecture. By adding new requirements one at a time without reconsidering the system as a whole at each step, you create the ideal conditions for a BBM.

In a RAD world, you tend to prefer code that just works and build applications by adding blocks that are mostly self-contained.

What common symptoms unequivocally tell you that you're going to have a bit too much mud in the gears of your system? A BBM doesn't get formed overnight and isn't that big in the beginning. Addressing these symptoms, which I'll describe, will help prevent it from growing too big.

The first alarm bell rings when you make a change in a class and end up breaking the code in another, apparently unrelated class. This is the nasty effect of rigid software, which is characterized by some level of resistance to change, which ultimately determines regression.

A second alarm bell rings when you fail in trying to reuse an apparently reusable piece of code. If the same code doesn't work once it's moved to another project, the most likely causes are

hard-to-find dependencies or tightly coupled classes. These are also the primary causes of software rigidity.

Tight coupling is beneficial because it helps you write code faster, and that code will likely run faster. It doesn't, however, make the code maintainable. In a project that grows piecemeal (like the vast majority of today's projects), maintainability is by far the primary software attribute you want to take into account.

Finally, a third alarm bell rings when you need to apply a fix to a function but don't feel confident in applying an ideal fix (or are unable to do so), so you resort to yet another workaround.

It took only a few years for .NET developers to be swamped with the same huge amount of complexity that Java colleagues experienced a decade earlier.

Any system can happily survive one occurrence (or even a few) of these symptoms. Their underlying mechanics are quite perverse, though. If you overlook one of these symptoms, you may incur the risk of making the system more convoluted.

For example, let's consider the second symptom (sometimes referred to as immobility). You have to add a new feature and you feel quite confident you can slightly adapt and reuse an existing function. You try and it doesn't work because the class you hoped would be reusable is, in reality, tightly connected to others. The alternative you have is importing a much larger set of functions in multiple modules. In such cases, a common (but naïve) solution is duplicating some code without ever attempting a cleaner reuse of classes. Duplication of code fixes the problem temporarily, but just grows the BBM.

Possible Causes of a BBM

It's rare for a single developer to create a BBM. Rather, a BBM has many causes, and often the primary cause has to be researched outside the current level of development. Demanding managers, as well as customers who don't really know what they want, convey to developers unclear and ambiguous information. Unless the team has plenty of experience in general software development and in the specific domain, this automatically leads to arbitrary choices successively fixed through compromises and workarounds. The net effect is that the overall architecture is weakened at its foundation.

Everybody involved in a software project can do a lot to avoid a BBM and to smooth its dramatic impact. Let's examine the fundamental steps to take when you find yourself immersed in a BBM.

A Disaster Recovery Strategy

When you face a BBM, the idealistic thing to do is simply rewrite the application based on reviewed requirements and new architectural choices. But a complete rewrite is never an option you're allowed to consider. How would you survive the mud?

Rock-Solid **REPORTING CONTROLS**

Optimized for .NET



Award-Winning Presentation Controls and Reporting Libraries



Learn more and download your **FREE** evaluation copy today
Visit **DEVEXPRESS.COM/REPORTING**
or Call Us (818) 844-3383

Devexpress™

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

All trademarks and registered trademarks are the property of their respective owners.

Foote and Yoder use an evocative image to introduce the only reasonable option you have in these cases: Sweep rubbish under the rug and go on with your own life. But I think I can summarize a recovery strategy for a software disaster in three steps. The first step is stopping any new development. The second step is isolating sore points to layers (when not tiered) and arranging tools to measure any regression on those points. Finally, you pick up each of those isolated problem layers and, with extreme care, try to refactor them to a cleaner and more loosely coupled architecture.

The second after you stop development on a muddy project, you start thinking of a bunch of relevant tests. Immersed in a BBM, the last thing you think of are classic unit tests. Relevant tests in this context are sort of integration tests that span multiple layers and sometimes tiers. These tests are slow to run in the first place, but—much more importantly—they may be slow to write. You need to isolate (probably large) chunks of behavior and hide them behind a façade that can be commanded through tests. Writing a façade may not be easy, but it's usually at least possible. It's mostly a matter of time and work. These tests are a big help for your next steps because they provide you with an automated tool to measure regression as you proceed with the final step. The final step consists of painstaking refactoring work where the first issue addressed is tight coupling.

Planning a Strategy to Prevent a BBM

Prevention is always preferable to treatment. There are three cardinal virtues of a team that may prevent a BBM stalemate, which I'll discuss later. Given that most projects these days suffer from a high level of requirements churn, piecemeal growth is a fact, not merely a possibility. To prevent a BBM, you must have an effective strategy to cope with piecemeal growth of functionality.

The mantra of maintainable software claims that perfect modern software serves the needs of today and is flexible enough to accommodate future requirements. The first virtue of the three mentioned earlier is domain experience. When you can't have a real domain expert on your team, at least you need someone with a deep understanding of the domain—which likely makes you the new domain expert. Domain experience takes you to sensible judgment about features that most likely will be required and requested. So, you can easily plan ahead while keeping the YAGNI (You Ain't Gonna Need It) principle clearly in mind.

As software units are maintained or changed, they become harder to further maintain or change.

The class-responsibility-collaboration (CRC) cards approach helped me to better understand the mechanics of domains I encountered for the first time. I see CRC cards more as a way to teach yourself the domain, than a way to come up with a proper design. Design, on the other hand, is proper only if validated by the actual customer. Use-cases—or, even better, a prototype—are smarter options.

The prototype can be a problem because customers may like it so much that they force you to extend it instead of starting from scratch with a new design. Prototypes are usually made with throwaway code deliberately written to be quick and, as such, devoid of principles and patterns. You just don't want to start with throwaway code.

The second virtue is becoming a better developer/architect by learning sane principles of software development and common best practices. The challenge here is all in learning how to do simple things correctly by default. Don't be afraid of interface-based programming and dependency injection. Validate each class against common OOP pitfalls. Ensure correctness via code and static analysis. If you're not sure about how a feature works, make it testable and write tests. Keep your code lean and mean, with most methods no longer than a few lines (with due exceptions, of course). If these practices become part of your tool chest, the BBM stays a bit farther away from your projects.

Everybody involved in a software project can do a lot to avoid a BBM and to smooth its dramatic impact.

The third virtue is about understanding the lifespan of the project. Not every project has to build a system that lasts for decades. Short-lived projects don't require the same design care. You can go faster on them and put care into making them work before making them right. In software, complexity must be controlled where it really exists, not created where it doesn't exist and isn't supposed to be. The danger, however, is when the initial belief that the project had a short time span is invalidated by surprising success and demand, and the creation of a market. If the subsequent release happens too slowly, you run the risk of a competitor snatching the market—or you run the risk of creating a BBM due to really tight (and desperate) business demands.

Software Project Disaster Recovery

The BBM is a common anti-pattern, but, to some extent, it's an attribute that can be applied to nearly every software project. It originates from using the wrong tools to tame complexity. In this article, I used an expression—disaster recovery—that's popular in IT. The mantra of disaster recovery experts, however, can be applied to software projects too: Dollars spent in prevention are worth more than dollars spent in recovery. Rush to your manager's office today! ■

DINO ESPOSITO is the author of "Programming Microsoft ASP.NET MVC3" (Microsoft Press, 2011) and coauthor of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2008). Based in Italy, Esposito is a frequent speaker at industry events worldwide. You can follow him on Twitter at twitter.com/despos.

THANKS to the following technical expert for reviewing this article:
Mircea Trofin

Powerhouse **ANALYTICS**

Optimized for .NET



Award-Winning Presentation Controls and Reporting Libraries



Learn more and download your **FREE** evaluation copy today
Visit **DEVEXPRESS.COM/ANALYTICS**
or Call Us (818) 844-3383

DevExpress[™]

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

All trademarks and registered trademarks are the property of their respective owners.



The Thread Pool Environment

The objects that make up the Windows thread pool API can be divided into two camps. In the first are those representing work, timers, I/O and waitable objects. These all potentially result in callbacks executing on the thread pool. I've already introduced the work object in last month's column and will explore the remaining objects in subsequent articles. In the second camp are those objects that control the environment in which these callbacks execute. That's the focus of this month's column.

The thread pool environment affects whether callbacks execute in the default pool or a specific pool of your own making, whether callbacks should be prioritized and so on. Being able to control this environment becomes increasingly important as you move beyond a handful of work objects or callbacks. It also reduces the complexity of coordinating the cancelation and teardown of these objects, the topic of next month's column.

The thread pool environment isn't an object in the same sense as the other objects that make up the thread pool API. For the sake of efficiency, it's simply declared as a structure so that you can directly allocate storage for it within your application. You should, however, treat it the same way as the other objects and not assume any knowledge of its internals, but rather access it only through the public set of API functions. The structure is named `TP_CALLBACK_ENVIRON`, and if you go and look it up you'll immediately notice that it has already changed since it was first introduced with Windows Vista. That's just another reminder that you must stick to the API functions. The functions themselves simply

manipulate this structure but shield you from any changes. They're declared inline to allow the compiler to optimize them as much as possible, so don't be tempted to think that you can do a better job.

The `InitializeThreadpoolEnvironment` function prepares the structure with default settings. The `DestroyThreadpoolEnvironment` function frees any resources used by the environment. As of this writing, it does nothing. This may change in future, however. Because it's an inline function, there's no harm in calling it, as it will just be compiled away. **Figure 1** shows a class that wraps this up.

The familiar `get` member function is provided for consistency with the `unique_handle` class template I introduced in my July 2011 column (msdn.microsoft.com/magazine/hh288076). Attentive readers may recall that the `CreateThreadpoolWork` and the `TrySubmitThreadpoolCallback` functions I introduced in last month's column had a final parameter that I didn't mention. I simply passed a null pointer value in each case. That parameter is actually a pointer to an environment, and is how you associate different work objects with an environment:

```
environment e;  
work w(CreateThreadpoolWork(callback, nullptr, e.get()));  
check_bool(w);
```

What good is this? Well, not much—that is, until you start customizing the environment.

Private Pools

By default, the environment will direct callbacks to the default thread pool for the process. This same thread pool would've handled the callbacks had you not associated the work with an environment. Consider what it means to have a default thread pool for a process. Any code running within the process can use this thread pool. Keep in mind that the average process loads dozens of DLLs directly or indirectly. It should be obvious that this can seriously impact performance. This isn't necessarily a bad thing. Sharing a thread pool among different subsystems within a process can often improve performance because the limited number of physical processors in the computer can be efficiently shared.

The alternative is that each subsystem creates its own pool of threads that all contend for processor cycles in a much more uncooperative manner. On the other hand, if a particular subsystem is abusing the default thread pool, you may want to shield yourself from this by using a private pool. This other subsystem may queue long-running callbacks or so many callbacks that the response time for your callbacks is unacceptable. You may also have specific requirements that necessitate certain limits on the number of threads in the pool. This is where the pool object comes in.

Figure 1 Wrapping the `InitializeThreadpoolEnvironment` Function

```
class environment  
{  
    environment(environment const &);  
    environment & operator=(environment const &);  
  
    TP_CALLBACK_ENVIRON m_value;  
  
public:  
  
    environment() throw()  
    {  
        InitializeThreadpoolEnvironment(&m_value);  
    }  
  
    ~environment() throw()  
    {  
        DestroyThreadpoolEnvironment(&m_value);  
    }  
  
    TP_CALLBACK_ENVIRON get() throw()  
    {  
        return &m_value;  
    }  
};
```

Optimized for .NET



Award-Winning Presentation Controls and Reporting Libraries



Learn more and download your **FREE** evaluation copy today
Visit **DEVEXPRESS.COM/CHARTING**
or Call Us (818) 844-3383



PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

All trademarks and registered trademarks are the property of their respective owners.

Figure 2 Defining the Serial_Pool Class

```
class serial_pool
{
    typedef concurrent_queue<function<void()>> queue;

    pool m_pool;
    queue m_queue, m_queue_high;
    work m_work, m_work_high;

    static void CALLBACK callback(
        PTP_CALLBACK_INSTANCE, void * context, PTP_WORK)
    {
        auto q = static_cast<queue*>(context);

        function<void()> function;
        q->try_pop(function);

        function();
    }
}
```

The `CreateThreadpool` function creates a private pool object completely independent from the default thread pool. If the function succeeds, it returns an opaque pointer representing the pool object. If it fails, it returns a null pointer value and provides more information via the `GetLastError` function. Given a pool object, the `CloseThreadpool` function instructs the system that the object may be released. Again, the `unique_handle` class template I introduced in my July 2011 column takes care of these details with the help of a pool-specific traits class:

```
struct pool_traits
{
    static PTP_POOL invalid() throw()
    {
        return nullptr;
    }

    static void close(PTP_POOL value) throw()
    {
        CloseThreadpool(value);
    }
};

typedef unique_handle<PTP_POOL, pool_traits> pool;
```

I can now use the convenient typedef and create a pool object as follows:

```
pool p(CreateThreadpool(nullptr));
check_bool(p);
```

I'm not hiding anything this time. The parameter in this case is reserved for future use and must be set with a null pointer value.

Figure 3 The Serial_Pool Constructor

```
public:

    serial_pool() :
        m_pool(CreateThreadpool(nullptr))
    {
        check_bool(m_pool);
        check_bool(SetThreadpoolThreadMinimum(m_pool.get(), 1));
        SetThreadpoolThreadMaximum(m_pool.get(), 1);

        environment e;
        SetThreadpoolCallbackPool(e.get(), m_pool.get());

        SetThreadpoolCallbackPriority(e.get(), TP_CALLBACK_PRIORITY_NORMAL);
        check_bool(m_work.reset(CreateThreadpoolWork(
            callback, &m_queue, e.get())));

        SetThreadpoolCallbackPriority(e.get(), TP_CALLBACK_PRIORITY_HIGH);
        check_bool(m_work_high.reset(CreateThreadpoolWork(
            callback, &m_queue_high, e.get())));
    }
}
```

The `SetThreadpoolCallbackPool` inline function updates the environment to indicate which pool callbacks should be directed to:

```
SetThreadpoolCallbackPool(e.get(), p.get());
```

In this way, work objects and any other objects created with this environment will be associated with the given pool. You could even create a few different environments, each with its own pool, to isolate different parts of your application. Just be careful to balance the concurrency between the different pools so that you don't introduce excessive scheduling with too many threads.

As I hinted at before, it's also possible to set minimum and maximum limits on the number of threads in your own pool. Controlling the default thread pool in this way isn't permitted because it would affect other subsystems and cause all kinds of compatibility problems. For example, I might create a pool with exactly one thread to handle an API that has thread affinity, and another pool for I/O completion and other related callbacks without any limits, allowing the system to adjust the number of threads dynamically as needed. Here's how I would set a pool to allocate exactly one persistent thread:

```
check_bool(SetThreadpoolThreadMinimum(p.get(), 1));
SetThreadpoolThreadMaximum(p.get(), 1);
```

Notice that setting the minimum can fail, whereas setting the maximum can't. The default minimum is zero and setting to anything else can fail because it will actually attempt to create as many threads as requested.

Another feature that the thread pool environment enables is the ability to prioritize callbacks.

Prioritizing Callbacks

Another feature that the thread pool environment enables is the ability to prioritize callbacks. This happens to be the only addition to the Windows Thread Pool API in Windows 7. Keep this in mind if you're still targeting Windows Vista. A prioritized callback is guaranteed to execute ahead of any callbacks with a lower priority. This doesn't affect thread priorities and therefore won't cause executing callbacks to be preempted. Prioritized callbacks simply affect the order of callbacks that are pending execution.

There are three priority levels: low, normal and high. The `SetThreadpoolCallbackPriority` function sets the priority of an environment:

```
SetThreadpoolCallbackPriority(e.get(), TP_CALLBACK_PRIORITY_HIGH);
```

Again, any work objects and other objects created with this environment will have their callbacks prioritized accordingly.

A Serial Pool

I introduced the `functional_pool` sample class in last month's column to demonstrate the various functions related to work objects. This time, I'm going to show you how to build a simple prioritized serial pool, making use of all the functions I've introduced this month that deal with the thread pool environment. By serial, I mean that I want the pool to manage exactly one persistent thread. And by prioritized, I'm simply going to support the submission of

Figure 4 Serial and Prioritized Behavior at Work

```
int main()
{
    serial_pool pool;

    for (int i = 0; i < 10; ++i)
    {
        pool.submit([])
        {
            printf("normal: %d\n", GetCurrentThreadId());
        };

        pool.submit_high([])
        {
            printf("high: %d\n", GetCurrentThreadId());
        };
    }
    getch();
}
```

functions at either normal or high priority. I can go ahead and start defining the `serial_pool` class, as shown in **Figure 2**.

Unlike the `functional_pool` class, the `serial_pool` actually manages a pool object. It also needs separate queues and work objects for normal and high priority. The work objects can be created with different context values pointing to the respective queues and then simply reusing the private callback function. This avoids any branching at run time on my part. The callback still just pops a single function off the queue and calls it. However, the `serial_pool` constructor (shown in **Figure 3**) has a bit more work to do.

First up is the creation of the private pool and setting its concurrency limits to ensure serial execution of any callbacks. Next, it creates an environment and sets the pool for subsequent objects to adopt. Finally, it creates the work objects, adjusting the environment's priority to establish the work objects' respective priorities and make the connection to the private pool that they share. Although the pool and work objects need to be maintained for the lifetime of a `serial_pool` object, the environment is created on the stack because it's only needed to establish the relationships between the various interested parties.

The destructor now needs to wait for both work objects to make sure no callbacks execute after the `serial_pool` object is destroyed:

```
~serial_pool()
{
    WaitForThreadPoolWorkCallbacks(m_work.get(), true);
    WaitForThreadPoolWorkCallbacks(
        m_work_high.get(), true);
}
```

And finally, two `submit` functions are required to queue functions at either normal or high priority:

```
template <typename Function>
void submit(Function const & function)
{
    m_queue.push(function);
    SubmitThreadPoolWork(m_work.get());
}
```

```
template <typename Function>
void submit_high(Function const & function)
{
    m_queue_high.push(function);
    SubmitThreadPoolWork(m_work_high.get());
}
```

Ultimately, it all comes down to how the work objects were created, in particular what information about the desired thread pool environment was provided. **Figure 4** shows a simple example that you can use, and in which you can clearly see the serial and prioritized behavior at work.

In the example shown in **Figure 4**, it's possible that one normal-priority callback will execute first—depending on how quickly the system responds—because it's submitted first. Beyond that, all of the high-priority callbacks should execute, followed by the remaining normal-priority ones. You can experiment by adding `Sleep` calls and raising the concurrency level to see how the thread pool adjusts its behavior according to your specifications.

Join me next month as I explore the critical cancelation and cleanup capabilities provided by the Windows thread pool API. ■

KENNY KERR is a software craftsman with a passion for native Windows development. Reach him at kennykerr.ca.

THANKS to the following technical expert for reviewing this article:
Stephan T. Lavavej

Data Quality Tools for .NET



IP Location



Property



International



Dedupe



Free Form
Parse



Address
Verification



Email
Validation



Name
Parse



Phone
Verification



Smart
Mover

Clean your database with tools that make it easy.

Request a free trial at
MelissaData.com/mynet or
Call 1-800-MELISSA (635-4772)

MELISSA DATA®
Your Partner in Data Quality



Second-Level Caching in the Entity Framework and AppFabric

The Entity Framework (EF) `ObjectContext` and `DbContext` maintain state information about entities they're managing. But once the context goes out of scope, that state information is gone. This type of caching is referred to as first-level caching and is only available for the lifetime of a transaction. If you're writing distributed applications using the EF where the context doesn't stick around—and therefore your state information isn't continuously available—the first-level cache likely won't suffice to support your demands. This is typically the case with Web applications and services—or even when you're using some type of repository pattern implementation where a long-running context isn't available.

Why the EF Can Benefit from Second-Level Caching

Why should you care about having access to a representation of the original state across processes? One of the great benefits of the EF is its ability to automatically generate database persistence commands (inserts, updates and deletes) based on the state information found in the context. But if that state information is unavailable, the EF has nothing to do when it's time to call `SaveChanges`. Developers, including myself, have been trying to work around this limitation since the EF was first introduced in 2006.

When executing a query against the context, you'll want to first see if that data exists.

Second-level caches are instrumental in solving this type of problem. These caches exist outside of the transaction—often outside of the application—and therefore are available to any context instance. And second-level caching is a commonly used coding pattern for caching data for various uses.

Rather than write your own way of caching data, there are caching mechanisms available such as memcached (memcached.org), and even caching support in Microsoft AppFabric (available in Windows Server as well as in Windows Azure). These services

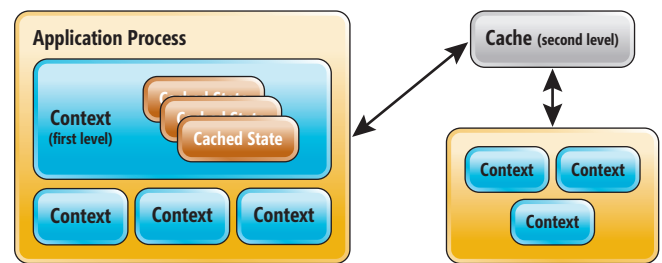


Figure 1 First-Level Caching Happens Inside a Transactional Context and Second-Level Caching Is External

provide the infrastructure for caching so you don't have to sweat the details. And they expose APIs that make it easy for programmers to read, store and expire data in the cache.

If you have a highly transactional system that can benefit from a cache to avoid repeatedly hitting the database for commonly queried data, you'll also find yourself looking for a caching solution. This is a great way to enhance performance when using data that's modified infrequently—for example, reference data or a list of players on a sports team.

Figure 1 shows the first-level cache maintained within an EF context, as well as various contexts accessing a common second-level cache.

Using the EF Caching Provider to Add Second-Level Caching

Designing the logic for reading, storing and expiring cache data takes a bit of work. You'd want to do this when working with the EF when you're querying for data or storing data. When executing a query against the context, you'll want to first see if that data exists in the cache so you don't have to waste resources on a database call. When updating data by using a context's `SaveChanges` method, you'll want to expire and possibly refresh the data in the

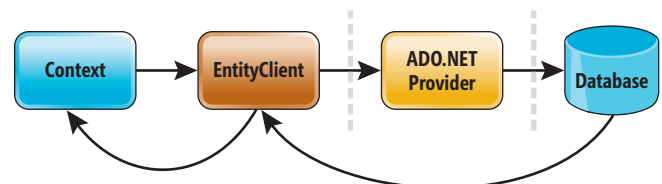


Figure 2 Flow from the EF Context Through an ADO.NET Provider to Get to the Database

This article discusses a prerelease version of Windows Azure AppFabric. All information is subject to change.

Code download available at code.msdn.microsoft.com/mag201109DataPoints.

LEADTOOLS® DOCUMENT SDKS



• PDF READ/WRITE/EXTRACT/VIEW/EDIT • OCR/ICR/OMR/MICR

• DOCUMENT READERS/WRITERS/CLEANUP/PRE-PROCESSING
• FORMS RECOGNITION/PROCESSING • ANNOTATIONS

• IMAGE FORMATS & COMPRESSION • SCANNING • VIEWER CONTROLS

• C++ • .NET • SILVERLIGHT • WINDOWS PHONE • CLOUD SDK
• C DLL • WCF SERVICES • WF ACTIVITIES • WPF • ASP.NET & COM

cache. And working with the cache is more complex than simply reading and writing data. There are plenty of other considerations to take into account. There's an in-depth article from the Association for Computing Machinery (ACM) that lays out the complexities of ORM caching, "Exposing the ORM Cache: Familiarity with ORM caching issues can help prevent performance problems and bugs" (bit.ly/k5bzd1). I won't attempt to repeat the pros, cons and hot points outlined in the article. Instead, I'll focus on implementation.

provider that captures messages between the Entity Client and the ADO.NET provider of choice (whether that's SqlClient, MySQL Connector or another) and injects logic to interact with a second-level caching mechanism. The wrapper is extensible. It provides underlying logic for any type of caching solution, but then you need to implement a class that bridges between this wrapper and the caching solution. The provider sample works with an in-memory cache, and the solution has a sample adapter

If you have a highly transactional system that can benefit from a cache to avoid repeatedly hitting the database for commonly queried data, you'll also find yourself looking for a caching solution.

The EF doesn't have built-in support for working with second-level caches. That functionality would make the most sense as part of theObjectContext and DbContext logic when they're about to interact with the database. But implementing the caching while taking into account the various issues discussed in the ACM article is non-trivial, especially with the lack of obvious extensibility points in the EF. One of the features that's frequently highlighted as a big difference between the EF and NHibernate is the fact that NHibernate has built-in support for implementing second-level caching.

But all is not lost! Enter the EF providers and the brainy Jarek Kowalski (j.mp/jkefblog), former member of the EF team.

The EF provider model is the key to how the EF is able to support any relational database—as long as there's a provider written for that database that includes EF support. There are a slew of third-party providers allowing you to use the EF with a growing array of databases (SQL Server, Oracle, Sybase, MySQL and Firebird are just some examples).

In the EF, theObjectContext talks to the lower-level EntityClient API, which communicates with the database provider to work out database-specific commands and then interacts with the database. When the database is returning data (as a result of queries or commands that update store-generated values), the path is reversed, as shown in **Figure 2**.

The spot where the provider lives is pliable, enabling you to inject additional providers between the EntityClient and the database. These are referred to as provider wrappers. You can learn more about writing ADO.NET providers for the EF or other types of providers on the EF team blog post, "Writing an EF-Enabled ADO.NET Provider" (bit.ly/etavcl).

A few years ago, Kowalski used his deep knowledge of the EF providers to write a

to use "Velocity," the code name for Microsoft distributed caching. Velocity eventually became the caching mechanism in the Microsoft Windows Server AppFabric.

Building an EFCachingProvider Adapter for Windows Server AppFabric

The EFCachingProvider was recently updated for the EF 4. The Tracing and Caching Provider Wrappers for Entity Framework

page (bit.ly/zlplb) includes great samples and documentation, so there's no need to repeat all of that here. However, the Velocity adapter was removed and there was no replacement to use the caching in AppFabric.

AppFabric lives in two places: Windows Server and Windows Azure. I've recreated the provider class that worked with Velocity so that it will now work with the caching in Windows Server AppFabric, and I'll share how to accomplish this yourself.

First, be sure you've installed the EF provider wrappers from bit.ly/zlplb. I've worked in the example solution, which contains the projects for the provider wrappers (EFCachingProvider, EFTracingProvider and EFProviderWrapperToolkit). There are also some client projects that test out the final caching functionality. The InMemoryCache provider is the default caching strategy and is built into the EFCachingProvider. Also highlighted in that project in **Figure 3** is ICache.cs. The InMemoryCache inherits from this, and so should any other adapter you want to create to use other caching mechanisms—such as the AppFabricCache adapter that I created.

In order to develop for AppFabric, you'll need the AppFabric cache client assemblies and a minimal installation of AppFabric on your development machine. See the MSDN Library topic, "Walkthrough: Deploying

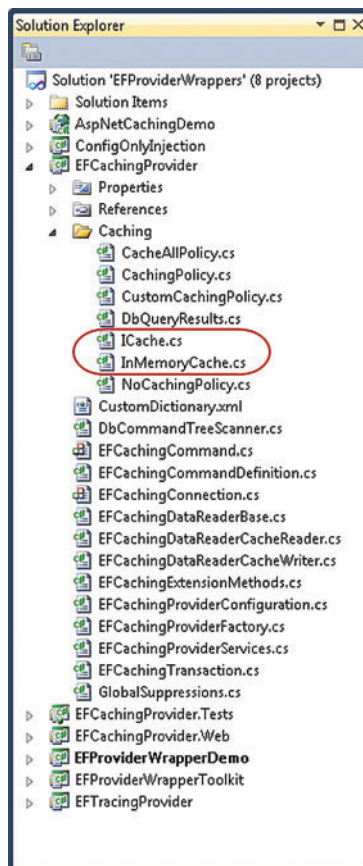


Figure 3 ICache and InMemoryCache Are Core Classes in the EFCachingProvider

WINFORMS • WPF • WEBFORMS • SILVERLIGHT
MVC • ACTIVEV • COMPACT FRAMEWORK

THE GREATEST FLEXGRID ON EARTH

"...I LOVE, LOVE, LOVE, the FlexGrid. It's like a Swiss army knife and I will never switch!"

—Darrin P. Dyson, Director of Development and Systems



	1	2	3	4	C
	Name				
1	<input type="checkbox"/>	Line: Computers (101 items)			
2	<input type="checkbox"/>	Color: Green (36 items)			
3	<input checked="" type="checkbox"/>	Rating: 4 (5 items)			
9	<input checked="" type="checkbox"/>	Rating: 2 (12 items)			
22	<input checked="" type="checkbox"/>	Rating: 3 (9 items)			
32	<input checked="" type="checkbox"/>	Rating: 0 (6 items)			
39	<input checked="" type="checkbox"/>	Rating: 1 (4 items)			
44	<input checked="" type="checkbox"/>	Color: Blue (28 items)			
78	<input checked="" type="checkbox"/>	Color: Red (22 items)			
106	<input checked="" type="checkbox"/>	Color: White (15 items)			
127	<input checked="" type="checkbox"/>	Line: Washers (77 items)			
229	<input checked="" type="checkbox"/>	Line: Stoves (72 items)			
261	<input checked="" type="checkbox"/>	Line: Dyers (70 items)			
289	<input checked="" type="checkbox"/>	Line: Microwaves (55 items)			
321	<input checked="" type="checkbox"/>	Line: Toasters (82 items)			

LIGHTWEIGHT!
PRINTING SUPPORT!
CELL MERGING!

SUPER FLEXIBLE!
EXCEL-LIKE EDITING!
UNBOUND MODE!

Experience The Magic @
COMPONENTONE.COM/FLEX



ComponentOne

Windows Server AppFabric in a Single-Node Development Environment,” at bit.ly/lwsolW, for help with this task. Be warned that it’s a bit involved. I’ve done it myself on two development machines.

Now you can create an adapter for Windows Server AppFabric. This is very close to the original Velocity3 adapter, but I did spend a bit of time learning how to work with the AppFabric client API in order to get these stars aligned. If you’re creating an adapter for a different caching mechanism, you’ll need to adjust accordingly to that cache’s API.

Another critical piece to the puzzle is to extend yourObjectContext class. I hope to try this out with an EF 4.1 DbContext soon, but this will necessitate modifying the underlying logic of the EFCachingProvider.

You use the same code to extend the context regardless of which implementation of ICache you’re working with. The extended class inherits from your context class (which, in turn, inherits from ObjectContext) and then exposes some extension methods from the EFCachingProvider. These extension methods enable the context to interact directly (and automatically) with the caching provider. **Figure 4** shows an example in the solution that extends NorthwindEFEntities, a context for a model built against the Northwind database.

I’ve added a Class Library project to the solution called EFAppFabricCacheAdapter. That project needs references to the

EFCachingProvider as well as two of the AppFabric assemblies: Microsoft.ApplicationServer.Caching.Core and Microsoft.ApplicationServer.Caching.Client. **Figure 5** shows my adapter class, AppFabricCache, which emulates the original VelocityCache.

The class uses the Microsoft.ApplicationServer.Caching.DataCache to fulfill the required implementation of ICache. Most notable is the use of AppFabric regions in the PutItem and InvalidateSets. When a new item is stored in the cache, the adapter also adds it to a region, or group, that’s defined by all entities in a particular entity set. In other words, if you have a model with Customer, Order and LinItem, then your Customer instances will be cached in a Customers region, Order instances in a region called Orders and so on. When a particular item is invalidated, rather than looking for that particular item and invalidating it, all of the items in the region are invalidated.

Another critical piece to
the puzzle is to extend your
ObjectContext class.

Figure 4 Extending an Existing Class that Inherits from ObjectContext, NorthwindEFEntities

```
using EFCachingProvider;
using EFCachingProvider.Caching;
using EFProviderWrapperToolkit;

namespace NorthwindModelDbFirst
{
    public partial class ExtendedNorthwindEntities : NorthwindEFEntities
    {

        public ExtendedNorthwindEntities()
            : this("name=NorthwindEFEntities")
        {
        }

        public ExtendedNorthwindEntities(string connectionString)
            : base(EntityConnectionWrapperUtils.
                CreateEntityConnectionWithWrappers(
                    connectionString, "EFCachingProvider"))
        {
        }

        private EFCachingConnection CachingConnection
        {
            get { return this.UnwrapConnection<EFCachingConnection>(); }
        }

        public ICache Cache
        {
            get { return CachingConnection.Cache; }
            set { CachingConnection.Cache = value; }
        }

        public CachingPolicy CachingPolicy
        {
            get { return CachingConnection.CachingPolicy; }
            set { CachingConnection.CachingPolicy = value; }
        }

        #endregion
    }
}
```

It’s this use of regions that caused me to set aside my attempt to implement Windows Azure AppFabric support. At the time that I’m writing this column, Windows Azure AppFabric is still a CTP and doesn’t support regions. Because underlying code of the caching provider is dependent on the regions and these methods, I was unable to easily create a provider implementation that would just work for Windows Azure AppFabric. You can, of course, call the InvalidateItem method yourself, but that would eliminate the benefit of the automated behavior of the provider.

Using the AppFabric Cache Adapter

There’s one last project to add, and that’s the project that exercises the adapter. The EFCachingProvider demo that’s part of the original solution uses a console app with a number of methods to test out the caching: SimpleCachingDemo, CacheInvalidationDemo and Non-DeterministicQueryCachingDemo. In my added console app for testing out the AppFabricCache, you can use the same three methods with the same implementation. What’s interesting about this test is the code for instantiating and configuring the AppFabricCache that will be used by the extended context in those three methods.

An AppFabric DataCache needs to be created by first identifying an AppFabric server Endpoint, then using its DataCacheFactory to create the DataCache. Here’s the code to do that:

```
private static ICache CreateAppFabricCache()
{
    var server = new List<DataCacheServerEndpoint>();
    server.Add(new DataCacheServerEndpoint("localhost", 22233));
    var conf = new DataCacheFactoryConfiguration();
    conf.Servers = server;
    DataCacheFactory fac = new DataCacheFactory(conf);
    return new AppFabricCache(fac.GetDefaultCache());
}
```

Note that I’m hardcoding the Endpoint details for the simplicity of this example, but you probably won’t want to do that in a production



Kendo

THE ART OF WEB DEVELOPMENT



HTML5 and jQuery. Taken a Step Further.

Build modern web apps for every browser and device with a pure client-side development toolset that simplifies HTML5 adoption. Enjoy quick development and deployment with a single, complete toolset built on jQuery. Rich web UI and the latest web standards made easy.



See the future of client-side development at
www.kendoui.com



Figure 5 Adapter that Interacts with Windows Server AppFabric

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using EFCachingProvider.Caching;
using Microsoft.ApplicationServer.Caching;

namespace EFAppFabricCacheAdapter
{
    public class AppFabricCache : ICache
    {
        private DataCache _cache;

        public AppFabricCache(DataCache cache)
        {
            _cache = cache;
        }

        public bool GetItem(string key, out object value)
        {
            key = GetCacheKey(key);
            value = _cache.Get(key);

            return value != null;
        }

        public void PutItem(string key, object value,
            IEnumerable<string> dependentEntitySets,
            TimeSpan slidingExpiration, DateTime absoluteExpiration)
        {
            key = GetCacheKey(key);
            _cache.Put(key, value, absoluteExpiration - DateTime.Now,
                dependentEntitySets.Select(c => new DataCacheTag(c)).ToList());

            foreach (var dep in dependentEntitySets)
            {
                CreateRegionIfNeeded(dep);
                _cache.Put(key, "", dep);
            }
        }

        public void InvalidateSets(IEnumerable<string> entitySets)
        {
            // Go through the list of objects in each of the sets.
            foreach (var dep in entitySets)
            {
                foreach (var val in _cache.GetObjectsInRegion(dep))
                {
                    _cache.Remove(val.Key);
                }
            }
        }

        public void InvalidateItem(string key)
        {
            key = GetCacheKey(key);

            DataCacheItem item = _cache.GetCacheItem(key);
            _cache.Remove(key);

            foreach (var tag in item.Tags)
            {
                _cache.Remove(key, tag.ToString());
            }
        }

        // Creates a hash of the query to store as the key
        private static string GetCacheKey(string query)
        {
            byte[] bytes = Encoding.UTF8.GetBytes(query);
            string hashString = Convert
                .ToBase64String(MD5.Create().ComputeHash(bytes));
            return hashString;
        }

        private void CreateRegionIfNeeded(string regionName)
        {
            try
            {
                _cache.CreateRegion(regionName);
            }
            catch (DataCacheException de)
            {
                if (de.ErrorCode != DataCacheErrorCode.RegionAlreadyExists)
                {
                    throw;
                }
            }
        }
    }
}
```

application. Once you've created the DataCache, you then use it to instantiate an AppFabricCache.

With this cache in hand, I can pass it into the EFCachingProvider and apply configurations, such as a DefaultCachingPolicy:

```
ICache dataCache = CreateAppFabricCache();
EFCachingProviderConfiguration.DefaultCache = dataCache;
EFCachingProviderConfiguration.DefaultCachingPolicy = CachingPolicy.CacheAll;
```

The EFProviderCache has been designed with extensibility in mind.

Finally, when I instantiate my extended context, it will automatically look for a caching provider, finding the AppFabricCache instance that I just set as the default. This will cause caching to be active using whatever configuration settings you applied. All you need to do is go about your business with the context—querying, working with objects and calling SaveChanges. Thanks to the

extension methods that bind your context to the EFProviderCache and the DataCache instance that you attached, all of the caching will happen automatically in the background. Note that the CacheAll CachingPolicy is fine for demos, but you should consider using a more fine-tuned policy so that you aren't caching data unnecessarily.

The EFProviderCache has been designed with extensibility in mind. As long as the target caching mechanism you want to use supports the standard implementations to store, retrieve, expire and group data, you should be able to follow the pattern of this adapter to provide a shared cache for your applications that use the EF for data access. ■

JULIE LERMAN is a Microsoft MVP, .NET mentor and consultant who lives in the hills of Vermont. You can find her presenting on data access and other Microsoft .NET topics at user groups and conferences around the world. She blogs at thedatafarm.com/blog and is the author of the highly acclaimed book, "Programming Entity Framework" (O'Reilly Media, 2010). Follow her on Twitter at twitter.com/julieleрман.

THANKS to the following technical experts for reviewing this article: Jarek Kowalski and Srikanth Mandadi

We didn't invent the Internet...

...but our components help you power the apps that bring it to business.



applications

powered by 

connectivity

powered by 

The Market Leader in Internet Communications, Security, & E-Business Components

Each day, as you click around the Web or use any connected application, chances are that directly or indirectly some bits are flowing through applications that use our components, on a server, on a device, or right on your desktop. It's your code and our code working together to move data, information, and business. We give you the most robust suite of components for adding Internet Communications, Security, and E-Business Connectivity to

any application, on any platform, anywhere, and you do the rest. Since 1994, we have had one goal: to provide the very best connectivity solutions for our professional developer customers. With more than 100,000 developers worldwide using our software and millions of installations in almost every Fortune 500 and Global 2000 company, our business is to connect business, one application at a time.

To learn more please visit our website →

www.nsoftware.com



Reporting on Diagnostics Data

You've probably been hearing a lot about Windows Azure lately. It's a great development platform for creating and deploying Web apps. Unfortunately, management can be a bit of a challenge. Windows Azure provides a nice framework for collecting and transferring diagnostic data about the running application and role, but falls short on getting the data into a useful data store—and it doesn't help with the visualization and analysis of that data.

Enter the SQL Azure Reporting CTP, our topic for this month. It may be mundane (by developer standards), but I'm going to cover it in the context of managing and monitoring my Windows Azure deployments. I'll demonstrate it using data from performance counters captured as part of the diagnostics for a Web Role. To date, the mechanism most folks use is to either transfer the information back home or access a SQL Azure data store from a local machine via reporting tools like SQL Server Reporting Services or Microsoft Excel. Unfortunately, current implementations following those paths have some significant downsides. On the one hand, if all the data is being transferred to a local store for reporting, this can both run up costs and present stale information. On the other hand, if you're accessing SQL Azure to run a local report or tool, you might decrease the amount of data transferred by aggregating and summarizing the data, but data transfer is still there—and with a nontrivial report, you'll likely still have some latency. By keeping the data in the cloud and also reporting on it from the cloud, you can send only the report view of the data, resulting in the freshest data and the least latency for viewing the report.

The Design and Setup

To get this project underway, the first thing I've got to do is determine what data I'll be collecting, where I'm going to put it and how I'm going to report on it. In this case, I'm going to collect percent time in GC and percent CPU time, both of which are common counters I look at when running performance tests. I'll need to move the data from its landing place in Azure Table Storage to SQL Azure in order to facilitate reporting. **Figure 1** illustrates the flow of data.

Using this diagram of the data flow, I can also identify my areas of effort. The numbers associate the following work items with the diagram:

1. Configure diagnostics for roles
2. Done through configuration

This article is based on a prerelease version of SQL Azure Reporting CTP. All information is subject to change.

Code download available at code.msdn.microsoft.com/mag201109Cloudy.

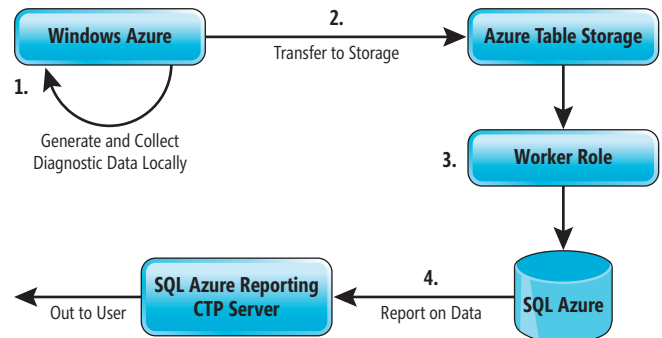


Figure 1 The Flow of Data

3. Transfer to SQL Azure
 - a. Create a worker role that at some frequency moves data
 - b. Define a schema for the data tables
4. Reporting
 - a. Create a query to retrieve needed data in a usable format
 - b. Create a report definition to render data into the report

SQL Azure Reporting

At the time of this writing, the reporting functionality of SQL Azure exists as a CTP that's available for sign-up at connect.microsoft.com/sqlazurectps. Once SQL Azure Reporting is active on my account, I can navigate to it in the management portal and see something like what's shown in **Figure 2**.

It's important to note that I'll need both the Web Service URL and the Username later on as I create this example. If you're following along, you'll need yours as well. Basically, once SQL Azure is provisioned, it's ready and simply awaiting deployment of your reports.

You'll also need Business Intelligence Development Studio (BIDS) to create the reports and deploy them to the service.

Diagnostics Setup

To collect the data I need, I have to create a couple of Performance-CounterConfiguration objects and add them to the Performance-Counters.DataSources in the OnStart method of the roles I'm interested in. I have a simple Web Role that's not modified from the template, but it will provide a source of data. I'll set both of my counters to sample every five seconds and then transfer the collected data to Table Storage every minute. **Figure 3** shows the code for this. These numbers are completely ridiculous to use in anything other than a proof-of-concept sample, but that's exactly what I've created, and I'd like to collect enough data in a short amount of time to generate something representative.



NetAdvantage® for jQuery

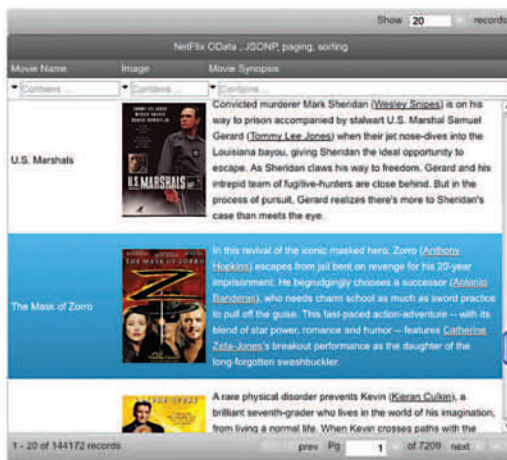
Product ID	Product Name	Product Number	Standard Cost
318	Adjustable Race	AR-5381	0
319	Searing Ball	BA-6327	0
320	3B Ball Bearing	BE-2349	0
321	Headset Ball Bearings	BE-2908	0
322	Blade	BL-2036	0
323	LL Crankarm	CA-5965	0
324	ML Crankarm	CA-6738	0
325	HL Crankarm	CA-7457	0
326	Channing Bolt	CB-2903	0
327	Channing Nut	CN-6137	0
328	Channing	CR-7633	0
329	Crown Race	CR-9981	0
330	Chain Stays	CS-2812	0
331	Decal 1	DC-8732	0
332	Decal 2	DC-9824	0
333	Down Tube	DT-2377	0

IG GRID ALL FEATURES ENABLED

The grid's sophisticated filtering and sorting features allow your users to find the records they need to work with.

IG GRID ON HTML PAGE

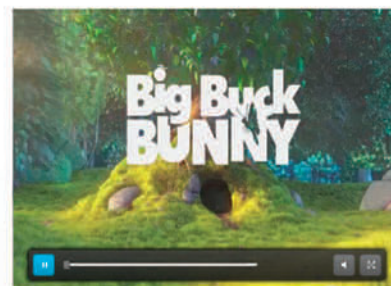
High performance grid lets users page through any OData source, even streaming movie descriptions.



Movie	Rating
The Shawshank Redemption (1994)	★★★★★★★★☆
Star Wars: Episode V (1980)	★★★★★★★★☆
Raiders of the Lost Ark (1981)	★★★★★★★★☆
The Lord of the Rings (2002)	★★★★★★★★☆
Taxi Driver (1976)	★★★★★★★★☆
Paths of Glory (1957)	★★★★★★★★☆
Star Trek (2009)	★★★★★★★★☆
Life of Brian (1979)	★★★★★★★★☆
Rocky (1976)	★★★★★★★★☆
Spartacus (1960)	★★★★★★★★☆

RATING

Enable a social experience for users, by letting them rate their favorite movies or anything else with our rating control.



IG VIDEO PLAYER MVC

When a user finds what they want to watch, our HTML5 video player adds streaming video right into your own apps.

IG EDITORS

Robust data entry and editing controls assist users with entering data in the formats your application needs.

First Name	Jack
Last Name	Black
Credit Card Number	1234 5678 9102
Expiration Date	02-01-2012
Country	da (Denmark)
Price information	
Price	kr 0.00
Quantity	5
Discount	2.00%
Total	kr 0.00
<input type="button" value="Submit"/>	

SCAN HERE
for an exclusive
look at jQuery!
www.infragistics.com/jq



UPLOADERS

Accept file uploads of any kind of content straight from your users with background file transfers.

TAKE YOUR WEB APPLICATIONS TO
THE NEXT LEVEL WITH OUR TOOLS
INFRAGISTICS.COM/JQUERY

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

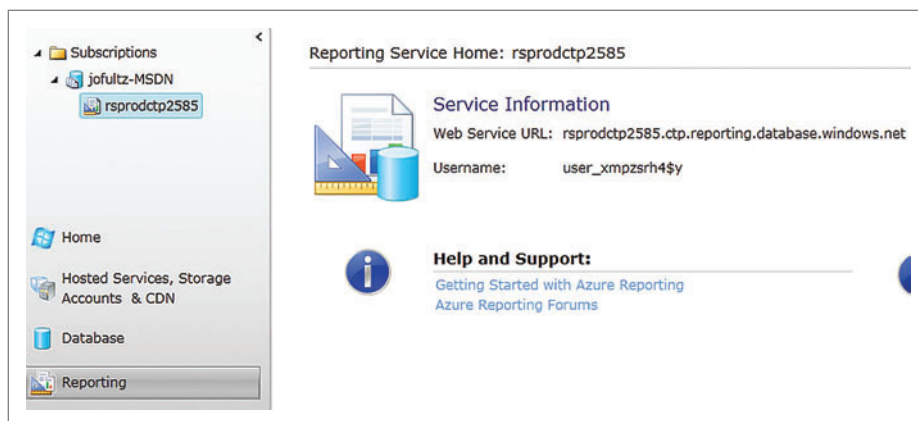


Figure 2 SQL Azure Reporting CTP

Capturing the Data and Reporting

Now I need to get the data into a format and a store for reporting, and then I can create the report. My obvious choice for storing the data will be SQL Azure, and not only because it's the structured storage solution for Windows Azure, it's also the only data source that SQL Azure Reporting can currently consume. When choosing the deployment location for the Windows Azure Storage, SQL Azure and SQL Azure Reporting roles, it's best to put them in the same datacenter or at least in the same geographic region to avoid any ingress or egress charges.

Data Schema and Transfer

For the example, I'll just keep the schema simple. I have an Id column, which is an identity to use as the primary key (PK), though a more realistic PK would be something like ComputerName + CounterName + TimeStamp. For now, I'll just use the identity field. **Figure 4** shows my simple schema for storing my performance counter data.

Figure 3 Sample Data Collection Code

```
// Create performance counter.
var performanceConfiguration = new PerformanceCounterConfiguration();
performanceConfiguration.CounterSpecifier = @"\Processor(_Total)\% Processor Time";
performanceConfiguration.SampleRate = System.TimeSpan.FromSeconds(5.0);

// Add counter to the configuration.
config.PerformanceCounters.DataSources.Add(performanceConfiguration);

performanceConfiguration = new PerformanceCounterConfiguration();
performanceConfiguration.CounterSpecifier =
@"\\.NET CLR Memory(_Global_)\% Time in GC";
performanceConfiguration.SampleRate = System.TimeSpan.FromSeconds(5.0);

config.PerformanceCounters.DataSources.Add(performanceConfiguration);

config.PerformanceCounters.ScheduledTransferPeriod =
System.TimeSpan.FromMinutes(1.0);

DiagnosticMonitor.Start(
"Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString", config);
```

Figure 4 Simple Schema for Storing Performance Counter Data

Name	Type
Id	Identity, Int
TimeStamp	Nvarchar(50)
CounterName	Nvarchar(50)
CounterValue	Real

I originally had the TimeStamp as an actual TimeStamp column, but I changed it to a string to make it easier to manipulate. As a matter of general practice, I use a local SQL Server instance for initial development and, when I'm ready, I script it to a new query window, change the connection to connect to SQL Azure, and run the script to create the tables.

Now that I have data being placed in Table Storage and also a place to put it in SQL Azure, I need the worker role to move the data, so I add a worker role to my solution. Within the OnStart

method, I add the standard configuration publisher code because I need to grab some information from settings, as shown in **Figure 5**.

Next, I add a method named TransferPerfDataToSql to the class. Within this method I'm going to grab the data available and average it. I have the Run method sleeping for about a minute, so there will be multiple entries for each counter. Because I'm not really interested in the full fidelity of the collected counters, I'm going to average them at each point of transfer. Thus, whatever the interval is for sampling the performance counters, I'm going to turn it into a single entry for a minute-and-15-second interval. For clarity in this sample code, I'll do this separately for each counter, though it could be done in one statement. **Figure 6** shows what it looks like to do it for one counter.

After I get the value, I need to remove the data from Table Storage. I could call the REST API directly, but in this case I just loop through the selected objects and pass each one to DeleteObject on the context:

```
foreach (PerformanceData perfdata in selectedData)
{
    context.DeleteObject(perfdata);
}

context.SaveChanges();
```

I repeat this code for the GC counter as well. Each value I collect needs to be added to my data store. I've created an .edmx file for the CounterData table, against which I write the code shown in **Figure 7** to add data to the table.

With this in place, the only thing left to do is to write the actual report—that most coveted of all tasks. (Yes, that is sarcasm.)

Figure 5 Standard Configuration Publisher Code

```
ServicePointManager.DefaultConnectionLimit = 12;

CloudStorageAccount.SetConfigurationSettingPublisher((configName, configSetter) =>
{
    configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));
    RoleEnvironment.Changed += (sender, arg) =>
    {
        if (arg.Changes.OfType<RoleEnvironmentConfigurationSettingChange>().
            Any((change) => (change.ConfigurationSettingName == configName)))
        {
            if(!configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))
            {
                RoleEnvironment.RequestRecycle();
            }
        }
    };
});
```



NetAdvantage®

for Data Visualization



INTERACTIVE DASHBOARDS

Build rich dashboards that visualize business data to empower decision makers.



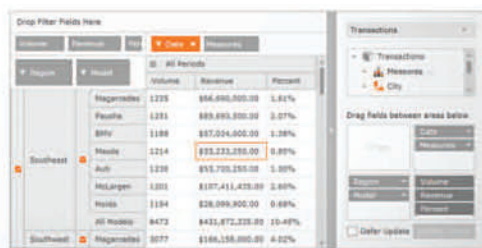
MEDIA TIMELINE

The timeline highlights how easily users can search, browse and play popular videos from YouTube.



MAP OUT ANYTHING AND EVERYTHING

Use xamMap™ to get an instant view and show anything including seating charts, floor plans, warehouse contents, and yes—geographic maps, too!



OLAP PIVOT GRID DATA VISUALIZATION

Work with multidimensional data from your OLAP cubes, data warehouses and Microsoft® SQL Server® Analysis Services.



INFRAGISTICS MOTION FRAMEWORK™

Create an immersive and animated user experience that tells the story of your data over time like no other visualization can.

SCAN HERE
for an exclusive
look at Data Visualization!
www.infragistics.com/dvis



TAKE YOUR APPLICATIONS TO
THE NEXT LEVEL WITH OUR TOOLS
INFRAGISTICS.COM/DV

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales 800 231 8588 • Infragistics Europe Sales +44 (0) 800 298 9055 • Infragistics India +91 80 4151 8042 • @infragistics

Copyright 1996-2011 Infragistics, Inc. All rights reserved. Infragistics and NetAdvantage are registered trademarks of Infragistics, Inc. Motion Framework and xamMap are trademarks of Infragistics, Inc. The Infragistics logo is a trademark of Infragistics, Inc.

Figure 6 Getting a Single Interval Value

```
var account = CloudStorageAccount.FromConfigurationSetting(
    "Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString");
var context = new PerformanceDataContext(account.TableEndpoint.ToString(),
    account.Credentials);
var data = context.PerfData;

// Get average CPU.
List<PerformanceData> selectedData =
    (from d in data
     where d.CounterName == @"\Processor(_Total)\% Processor Time"
     select d).ToList<PerformanceData>();

double AvgCPU = (from d in selectedData
                 where d.CounterName == @"\Processor(_Total)\% Processor Time"
                 select d.CounterValue).Average();
```

The Report

I'm not a particularly accomplished report writer and, to be quite honest, that's a characteristic I want to be sure to consistently demonstrate so as to never end up with that part of the project myself. So, I'll create a relatively simple report to illustrate displaying the data in a grid and plotting a line to visualize the data.

The first step is to query the data in the format I need in order to produce the report. The data I've stored in the SQL Azure database is the average value for about a minute of samples, which means that each data point represents a minute. For my report, however, I'd rather have the data as a single point for each hour of time, giving me 24 points per day that I might use to trend over a short number of days. To do this I'll tease apart the TimeStamp field using DATEPART and group by both day and hour. This results in the following query:

```
SELECT
    DATEPART(DD, [TimeStamp]) as [Day]
    ,DATEPART(HH, [TimeStamp]) as [Hour]
    ,[CounterName]
    ,Avg([CounterValue]) [Value]
FROM [dbo].[CounterData]
Group by DATEPART(DD, [TimeStamp]), DATEPART(HH, [TimeStamp]), CounterName
Order By CounterName
```

Moving on, I open BIDS and create a new report project. I add two Data Sources, one for cloud and one for local. It's best to develop the report against a local store, as it will be faster to preview the report, which means no egress fees will be accrued for report

Figure 7 Adding Data to the CounterData Table

```
PerfDataEntities pde = new PerfDataEntities();
CounterData cd = new CounterData();
cd.CounterName = @"\Processor(_Total)\% Processor Time";
cd.CounterValue = (float)AvgCPU;
cd.TimeStamp = System.DateTime.Now.ToString();

pde.AddToCounterDatas(cd);

cd = new CounterData();
cd.CounterName = @"\NET CLR Memory(_Global_)\% Time in GC";
cd.CounterValue = (float)AvgTimeGC;
cd.TimeStamp = System.DateTime.Now.ToString();

pde.AddToCounterDatas(cd);
pde.SaveChanges();
context.SaveChanges();
```

execution. As a matter of practice when working on a report—or on any project for the cloud—when I get a piece of it working locally I run a quick litmus test in the cloud to make sure that it works there as well. Having both data sources defined makes this pretty simple to test against SQL Azure, and I'll test major report sections by actually deploying it and running it from the SQL Azure Reporting CTP. After adding the data sources, I add a dataset to the report and use the previously mentioned SQL to define the data set.

With the supporting elements in place in the report project, I add a blank report to the project and name it CounterDataByHour.rdl. Because I want a grid of data and a line graph, I add a tablix and a sparkline chart. For the tablix I use the day value for the second through n column headers on the first row, the hour value for the second through n column headers on the second row, and the data as the values across the columns for each row generated by Group By CounterName. For the chart data I set the CounterName as the series data and use hour and day as the category groups. This results in a design surface that looks like what's shown in **Figure 8**.

There are expressions in the tablix, but those are merely CSTR functions on the field values. The report is ready to go and all I need to do is push it to the cloud. First, though, I need to configure the solution to deploy to the cloud. To do that, I open the solution properties and set the TargetServerURL to the URL provided in the Windows Azure Management Portal with "reportserver" tacked on to it. Thus, mine is: <https://rsprodctp2585.ctp.reporting.database.windows.net/ReportServer>.

I must use HTTPS for this operation for SQL Azure Reporting, but that's preferred in this case anyway because a company would want to keep its server state and performance information to itself. With this set I deploy the report and I'm prompted for my credentials—the username I copied from the Management Portal, as shown in **Figure 9**.

If nothing fails, the report should deploy successfully and I can navigate to the site directly, log in and run the report from the browser (**Figure 10**).

In some cases you'll want to link directly to the report, but in many

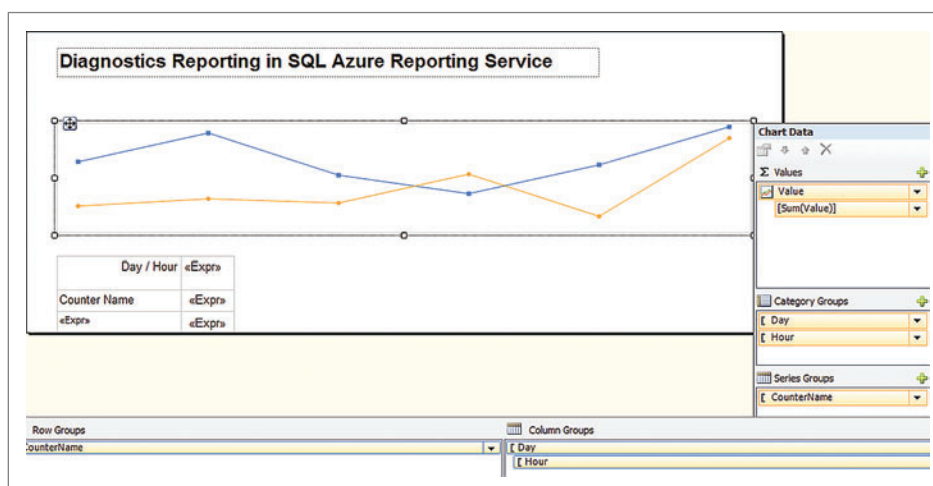
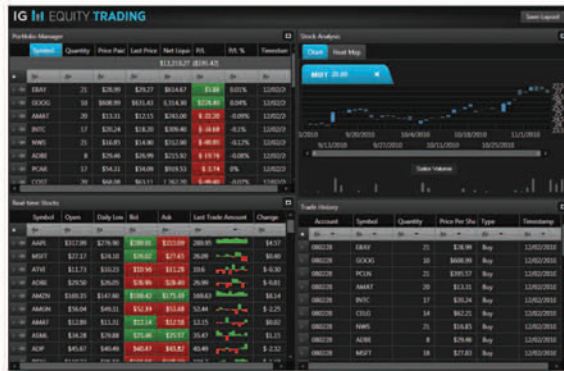


Figure 8 Designing the Chart

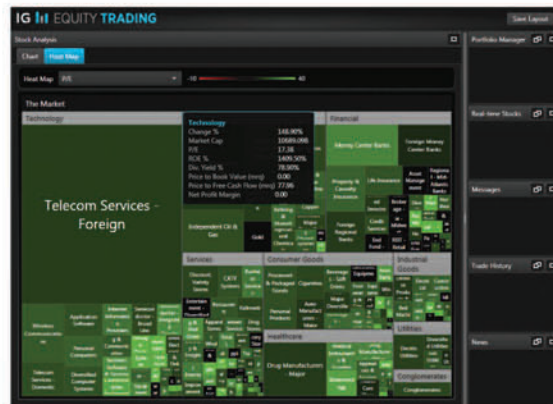
NetAdvantage[®] PERFORMANCE



**REAL-TIME
USER
INTERFACES**
Refresh your app's
user interface tick-
by-tick on every
value change in the
fastest applications
imaginable.



ACROSS ALL PLATFORMS
Charts and grids featuring blazing speed
whether you are using ASP.NET, Silverlight,
WPF or any other .NET platform.



DEEP DRILLDOWN
Dive into the deepest details of your
data with crisp, rapidly-updating visual
controls designed to handle it all.



TAKE IT ON THE ROAD
Our XAML controls with Windows Phone[®] 7
support means your UI is always on the go.



SCAN HERE
for an exclusive
look at Performance!
www.infragistics.com/perf

TAKE YOUR APPLICATIONS TO
THE NEXT LEVEL WITH OUR TOOLS
INFRAGISTICS.COM/PERFORMANCE

INFRAGISTICS[™]
DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales 800 231 8588 • Infragistics Europe Sales +44 (0) 800 298 9055 • Infragistics India +91 80 4151 8042 • info@infragistics.com

Copyright 1996-2011 Infragistics, Inc. All rights reserved. Infragistics and NetAdvantage are registered trademarks of Infragistics, Inc. The Infragistics logo is a trademark of Infragistics, Inc.

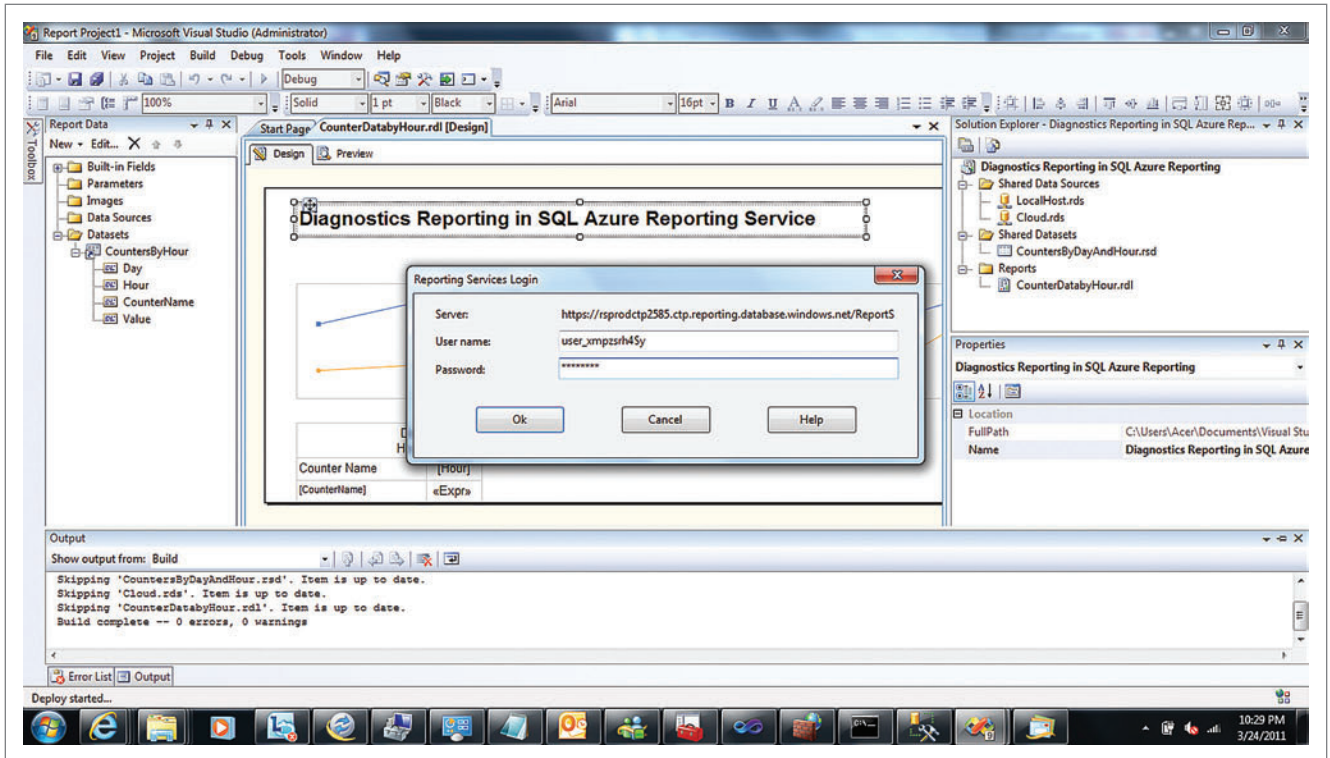


Figure 9 Deploying the Report

others you might use a report control embedded within an application or a Web page. The key point here is that for the person consuming the reports, nothing is different from the typical experience of using SQL Server Reporting Services in his solution. The report developer performs some slightly altered operations, but it's mostly the same for him, too.

Wrapping Up

My simple example is just that, but those who are a little more handy with BIDS and general report writing will find that the goodies in

the SQL Azure Reporting CTP—for creating parameters, linked reports, expressions and other such report-writing features—provide a platform for rich reports that are about as complex as anyone would want them to be. SQL Azure Reporting gives developers a familiar paradigm for developing and distributing reports, with the essential difference that no one has to worry about the infrastructure to host it. One of the biggest benefits is using it for reporting on various types of diagnostic data, as I did here with the performance counters, and also for monitoring various aspects of any deployment. With SQL Azure Reporting, you can

collect data to a central database for reporting and even report across deployments, storing data in separate databases—as long as they're SQL Azure databases. Other data sources are on the roadmap, but even in its current incarnation, SQL Azure Reporting is a powerful and welcome addition the Windows Azure toolbox. ■

JOSEPH FULTZ is a software architect at HP, working as part of the HP.com Global IT. Previously he was a software architect for Microsoft working with its top-tier enterprise and ISV customers defining architecture and designing solutions.

THANKS to the following technical expert for reviewing this article:
Jim Keane

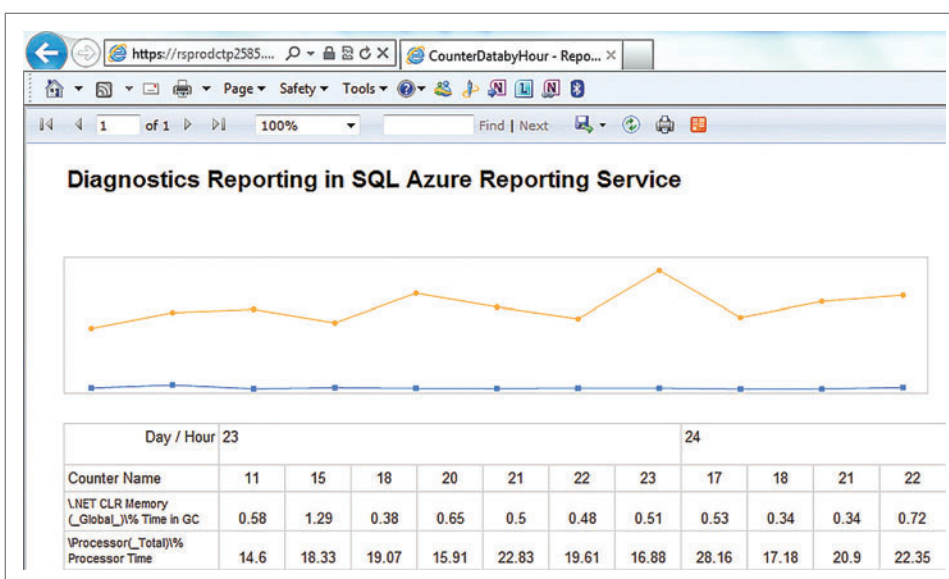


Figure 10 Viewing the Report in the Browser

Devart ORM Solutions



Create model
from DB or
DB from model



Synchronize
model and DB
in either direction



Adjust your
model
conveniently



Generate code
and mapping

Visual ORM Model Designer

**Entity Framework
LINQ to SQL
NHibernate**

Database-Optimized Connectivity



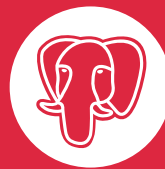
SQL Server



Oracle



MySQL



PostgreSQL



SQLite



Entity Developer

Powerful ORM model designing and code generation tool for Entity Framework, LinqConnect, LINQ to SQL, and NHibernate.



Linq Connect

Fast and easy-to-use ORM solution, developed closely to the Microsoft LINQ to SQL technology and supporting SQL Server, Oracle, MySQL, PostgreSQL, and SQLite.



dotConnect

Enhanced database connectivity solutions built over ADO.NET architecture with support for Entity Framework, LinqConnect, and NHibernate.

The 'Juneau' Database Project

Jamie Laflen and Barclay Hill

SQL Server Developer Tools, or SSDT (code-named “Juneau”), represents Microsoft’s continued commitment to providing integrated tools for developers targeting Microsoft SQL Server. Those familiar with the previous versions of the Database Project in Visual Studio will find that Juneau is an evolution of those tools for SQL Server, plus many new capabilities and improvements. Juneau provides a unified toolset that combines the projects found in SQL Server Business Intelligence Design Studio (BIDS)—including Reporting, Analysis and Integration Services projects—with the SQL Server Database Project.

While you can install Juneau independently and have everything you need to develop databases for SQL Server, it’s also an integral part of Visual Studio, meaning you can now perform your database development in the same environment as your application

development. Juneau is available as part of the SQL Server 2011 release (code-named “Denali”), but will also be available with future versions of Visual Studio.

This article focuses on the Juneau Database Project. This project system and its related features provide the tools to edit, compile, refactor and publish databases to specific versions of SQL Server and SQL Azure. In Juneau, there’s one database project for all versions of SQL Server, and it can include both Transact-SQL (T-SQL) scripts and code that defines SQL CLR objects. Let’s start by taking a look at setting up a database project.

The Database Project

The Database Project is a Visual Studio project enabling offline development of SQL Server databases. A database development team would move to project-based development to enjoy the benefits that this type of development affords the team over developing against a shared live database, including:

- Isolation of developer changes
- Rich T-SQL editing support
- Verification of source prior to deployment and enforcement of team coding standards through code analysis rules
- Automated migration-script generation

The core project system is very similar to its Visual Studio Database Project (.dbproj) predecessor, in which developers express objects declaratively as CREATE statements. For some additional background regarding offline schema development, see bit.ly/raDMNx.

This article is based on a prerelease version of SQL Server Developer Tools. All information is subject to change.

This article discusses:

- Offline database development
- Creating a database project
- SQL Server Express LocalDB
- Connected development

Technologies discussed:

SQL Server, SQL Server Developer Tools, SQL Server Database Project

Meet the EXPERTS



James Johnson
Product Manager



WE ARE REPORTING

- ✓ ActiveReports the award-winning .NET product allows Microsoft Visual Studio developers to provide royalty-free reporting solutions.
- ✓ ActiveReports now supports Silverlight and Azure.
- ✓ ActiveReports features a fast and flexible reporting engine.
- ✓ ActiveReports has an event-driven API to completely control the rendering of reports; Windows Forms, Silverlight, Web viewer controls and an end user designer control.
- ✓ ActiveReports has a wide range of export formats including Excel, TIFF, and PDF.
- ✓ ActiveReports multi-language support, extensive customization, and XCopy deployment simplifies Windows and Web reporting.

 **GrapeCity PowerTools**

www.GCPowerTools.com

GvTv.GCPowerTools.com

YOUR ONLINE SOURCE FOR DEVELOPER NEWS



GvTv.GCPowerTools.com

Smarter Components for Smarter Developers

GrapeCity PowerTools www.GCPowerTools.com

Setting Up a Project

When you first create a database project, it appears empty, like most other projects created by Visual Studio. This means you can add source code and then build and debug prior to checking it into source code control. Unlike most other projects where you must start from source code, a database project can be created from an existing SQL Server database. There are several ways to populate a project with source code, depending on the level of control you want:

- Import a complete database
- Import scripts
- Import a database package (.dacpac)
- Write updates for specific objects from a comparison of a project with a database
- Drag and drop from Server Explorer
- Convert an existing Visual Studio 2010 Project (Database/SQL CLR) into SQL Server Database Project

The project you create models a database; the database properties (for example, collation) are stored within the project's properties and user objects are stored as source within the project. The database project is the offline representation of your database in source code form.

To introduce the database project, let's start with a new empty database project and walk through its various features. There are several ways to create a database project. Because we'll start with an empty project, just click File | New | Project and then select the SQL Server Database Project, as shown in Figure 1.

Adding Source to Your Project

Most databases have at least one table, so let's add one now. SQL Server Database Project provides default T-SQL templates for many of the commonly used SQL objects. To create a new object, right-click the project node and select Add | Table, then specify the name of the table in the dialog; we'll use "Customer."

Figure 2 shows what a new table looks like in the new table designer.

Like the HTML designer, the table designer defaults to a split-pane view. The designer window contains a graphical representation of the

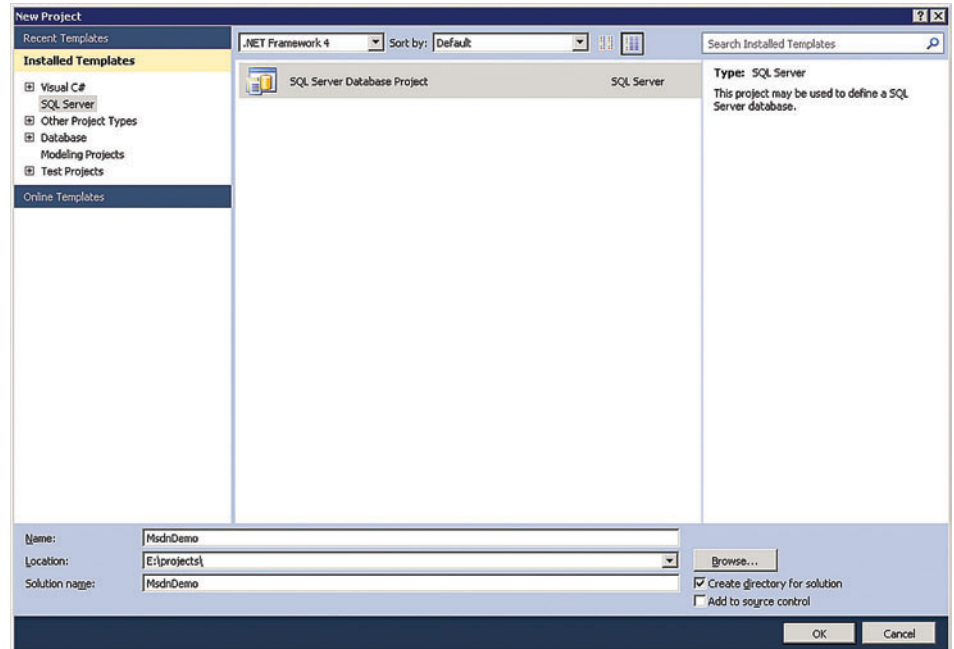


Figure 1 Creating a New SQL Server Database Project

table as well as the table's script definition. No matter which view you choose to work in, your changes will be synchronized with the other view.

Often, direct access to a table is restricted and you'll want to write a stored procedure to give an application programmatic access to the table's data. With this in mind, we'll add a stored procedure called "CreateCustomer" the same way we added the Customer

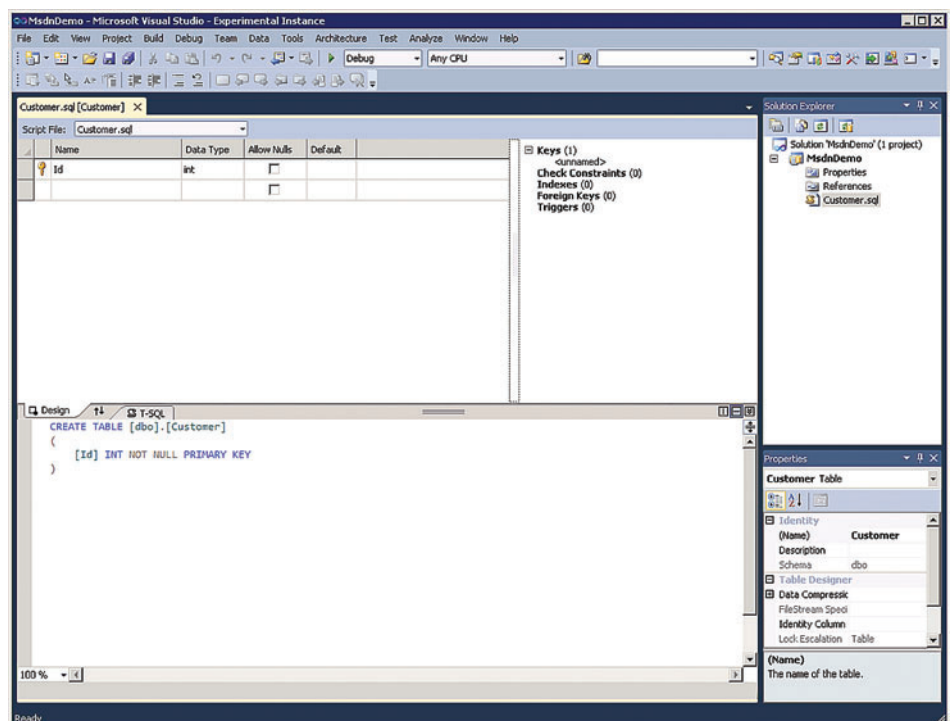


Figure 2 The New Table Designer

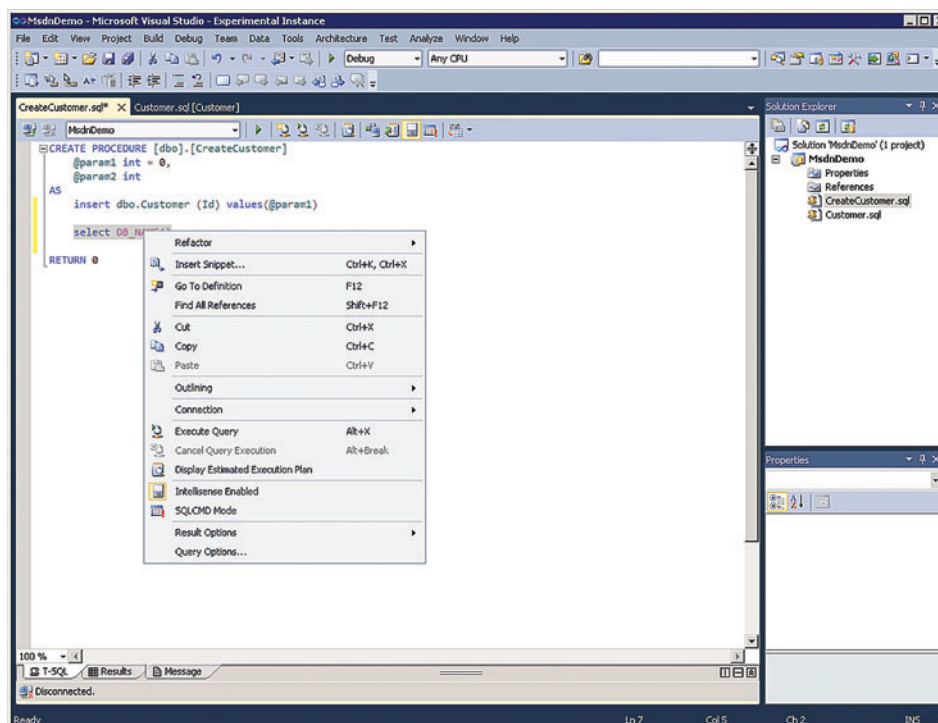


Figure 3 Executing Project Source Code Against the Debug Database

table. The editor opens and you're presented with a default procedure skeleton to populate. This might seem a little daunting because, for many developers, you develop your stored procedures against a live database so you can execute and validate your code as you write it. But don't worry; the database project creates a new debug database to make project-based development much more productive. Within the editor you can select some text and right-click to get the context menu shown in **Figure 3**.

Project. A typically unsung hero, IntelliSense inside the project system has been improved based on both SQL Server Management Studio (SSMS) and the Database Project feedback. Many of the changes just smoothed out previous rough edges, but a few are significant:

- Preview mode: One of the biggest complaints about T-SQL IntelliSense centered on unintended completions when referencing yet-to-be-defined types. For example, you may have wanted to type:

How does executing the text using Execute Query, without having configured a connection string, work? When any source code is opened, the editor is configured to execute against the debug database when asked to execute the script or selected text. In addition, the debug database is updated when you start debugging (for example, by hitting F5 or Ctrl+F5) to match the source of your database project. Juneau leverages a new feature in Denali called the SQL Server Express LocalDB to provide an instance and databases automatically to develop and debug against. For more information, see "Introducing SQL Server Express LocalDB" on p. 36.

Developing Your Source

Once you've looked at LocalDB and have seen how you can develop stored procedures within your source code, you'll want to know about some other cool features available in the Database

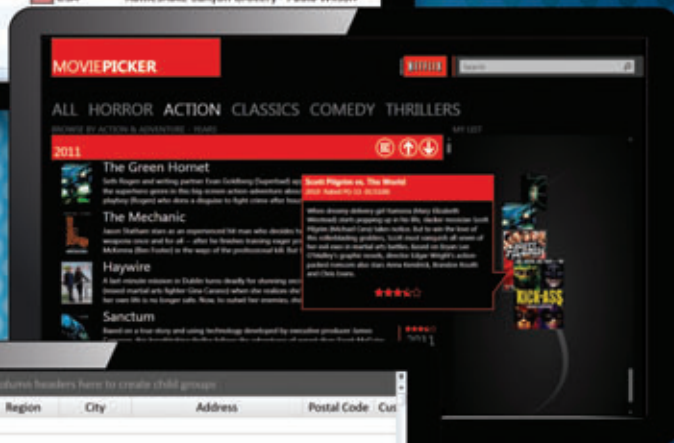
Figure 4 Options for Verifying Your Code

If You Are Here ...	Do the Following ...
You've made several changes and want to debug.	<ol style="list-style-type: none"> 1. Set the database project to be the startup project. 2. Add a script to the project and write the test case that you want to debug. 3. Open the project properties, go to the debug tab and select the script as the Startup script. 4. Hit F5 to push your changes to your debug database (default LocalDB) and start the debugger. 5. The debugger will break on the first line of the Startup script.
You have an application project (Web application or .exe) that uses stored procedures that you've changed (and may want to debug).	<ol style="list-style-type: none"> 1. Set the application project to be the startup project. 2. Modify the application's configuration file (web.config or app.config) to point to the debug database (by default Data Source=(localdb)\<SolutionName>; Initial Catalog=<Project Name>). 3. Hit F5 to run the application and update the debug database. 4. Interact with the application to test. 5. Put a breakpoint in the stored procedure you want to debug.
You want to debug your database using an application that accesses your debug database.	<ol style="list-style-type: none"> 1. Set the database project to be the startup project. 2. Modify the application's configuration file to point to the debug database. 3. In the database project's properties Debug tab, specify to run the application as an external program. 4. Hit F5. 5. Put a breakpoint in the stored procedure you want to debug. 6. Interact with the application that was started when you hit F5.
You've made several changes and you want to test (and possibly debug) the changes.	<ol style="list-style-type: none"> 1. Add a script (not in build) to the project and write the tests you want to perform. 2. Hit Ctrl-F5 to push your changes to your debug database (default LocalDB). 3. Highlight each test and right-click execute (or execute with debug) to verify that the results are what you expect.

THESE GUYS STAND ON THEIR OWN.

That's because we focus on the most important controls, not dozens of generic, bundled ones.

ID	Employee	Country	Customer
USA (122 items)			
Albuquerque (18 items)			
11,077	Davolio, Nancy	USA	Rattlesnake Canyon Grocery - Paula Wilson
11,000	Fuller, Andrew	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,988	Leverling, Janet	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,889	Dodsworth, Anne	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,852	Callahan, Laura	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,820	Leverling, Janet	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,761	Buchanan, Steven	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,598	Davolio, Nancy	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,569	Buchanan, Steven	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,564	Peacock, Margaret	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,479	Leverling, Janet	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,401	Davolio, Nancy	USA	Rattlesnake Canyon Grocery - Paula Wilson
10,346	Leverling, Janet	USA	Rattlesnake Canyon Grocery - Paula Wilson



Order ID	Country	Region	City	Address	Postal Code	Customer
USA						
CO						
FL						
10310	USA	FL	Jacksonville	89 Jefferson Way Suite 2	97201	77
10317	USA	FL	Jacksonville	89 Chiaroscuro Rd.	97219	48
10805	USA	FL	Miami	89 Jefferson Way Suite 2	97201	77
10867	USA	FL	Jacksonville	89 Chiaroscuro Rd.	97219	48
10883	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
10992	USA	FL	Jacksonville	89 Jefferson Way Suite 2	97201	77
11018	USA	FL	Jacksonville	89 Chiaroscuro Rd.	97219	48
11169	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
11172	USA	FL	Jacksonville	89 Jefferson Way Suite 2	97201	77
11179	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
11406	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
11524	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48
11729	USA	FL	Jacksonville	89 Chiaroscuro Rd.	97219	48
11854	USA	FL	Jacksonville	89 Jefferson Way Suite 2	97201	77
11880	USA	FL	Miami	89 Chiaroscuro Rd.	97219	48



XCEED
DataGrid
for WPF

Mature, feature-packed, and lightning-fast. The most adopted and trusted WPF datagrid around!



XCEED
DataGrid
for Silverlight

The only Silverlight datagrid on the market with fast remote data retrieval and a completely fluid UI!



XCEED
Ultimate ListBox
for Silverlight

The same data virtualization and smooth-scrolling as our Silverlight datagrid, packed in the streamlined format of a listbox.

Try them live at
xceed.com

XCEED
MULTI-TALENTED COMPONENTS


```
select t.id from Table1 as t
```

But pressing a period after you typed:

```
select t
```

would have resulted in:

```
select Table1.
```

This problem has been fixed by adding “preview mode” to IntelliSense. With preview mode, users will see a set of completions, but they won’t get auto-complete behavior until they “activate” the completions (via either the down-arrow or up-arrow key).

- IntelliSense with unsaved files: In earlier database projects, you needed to save a file before the objects defined within were available to IntelliSense. For example, previously a newly added table would not appear in IntelliSense until the table’s file was saved. Now, the table will appear in IntelliSense even if the file is not saved.

When planning Juneau, the team felt it was important to elevate the T-SQL development experience to the level enjoyed by developers in other managed languages. As part of this goal, the editor inside the project system has been enhanced to provide common language features like GOTO definition, find all references, and refactoring directly from the editor by right-clicking on an object definition or reference.

As you know, refactoring databases is a lot harder than refactoring application code because databases have data and a seemingly innocuous name change can amount to many additional changes due to dependencies from other objects—or, worse still, data loss can occur when these changes are deployed. Juneau tracks refactor operations performed inside the project by tracking the changes in the `MsdnDemo.refactorlog` (in our case) so that it can take those operations into account and generate the appropriate `sp_rename` or `ALTER SCHEMA` statements.

In other managed languages, the act of building your project is the final step before running the new code. In database development, the analog would be creating all the database objects in your project within a running instance. To bring full database verification

Figure 5 Options for Migrating Your Code

Migration Scenario	Description
Promote changes to debug database	<ul style="list-style-type: none">• Quick update of your debug database with changes from your project• Uses settings from the Database Project Debug tab• Executed when user hits F5/Ctrl+F5• Exposed as an MSBuild target
Publish changes to a separate database	<ul style="list-style-type: none">• Formal update of a database from a project• Meant to migrate or update an environment• Executed by clicking on the project’s Publish menu• Exposed as an MSBuild target and through the command-line tool (<code>sqlx.exe</code>)• Exposed as a Web Deploy provider (<code>bit.ly/qBXOLK</code>)
Compare schema and move changes between databases	<ul style="list-style-type: none">• Visual differencing and update tool• Allows individual selection of objects to be updated• Takes project’s refactor log into account when displaying differences• Executed by selecting context menus on a project or database in the new SQL Server node of Server Explorer

Introducing SQL Server Express LocalDB

SQL Server Express LocalDB (or LocalDB for short) is

basically the next generation of SQL Express User Instances, but without the need to explicitly manage a SQL Express instance on your desktop. LocalDB doesn’t have a background service hosting a named instance; rather, it provides a way for developers to define custom named instances and then interact with them. When a LocalDB instance is started, it runs as a process under the credentials of the user who started it. For example, if you start a LocalDB instance, in Task Manager you’ll see a `sqlservr.exe` process running under your own credentials. This by itself is cool because it means no setup to debug your T-SQL code! Once the instance is started, it’s just like a SQL Express instance. When the instance isn’t used for a period of time, it will shut down so that an idle SQL Server instance doesn’t consume resources on your machine.

A LocalDB installation comes with a command-line tool that can be used to manage the LocalDB instances you’ve created. For example, you can create a new LocalDB instance called `LocalDBDemo` by executing the following:

```
C:\Program Files\Microsoft SQL Server\110\Tools\Binn>SqlLocalDB.exe
create LocalDBDemo
Local Database instance "LocalDBDemo" created with version 11.0.
```

Once the instance has been created, you can start it using the command-line tool or you can just attempt to connect to the instance through Juneau, SQL Server Management Studio (SSMS) or your application. Because LocalDB instances aren’t resident instances, they can’t be accessed using the `(local)` prefix used to address instances on the local machine. Instead, use `(localdb)`; in this example you’d type `(localdb)\LocalDBDemo` into Juneau to connect to and manage the instance.

When you create a new instance, a new directory is created within your user profile and the four built-in databases (`master`, `tempdb`, `msdb` and `model`) are placed within this directory. By default, any new databases will go in the instance’s directory as the default data path. In our example, the directory is:

```
C:\Users\User1\AppData\Local\Microsoft\Microsoft SQL Server Local
DB\Instances\LocalDBDemo
```

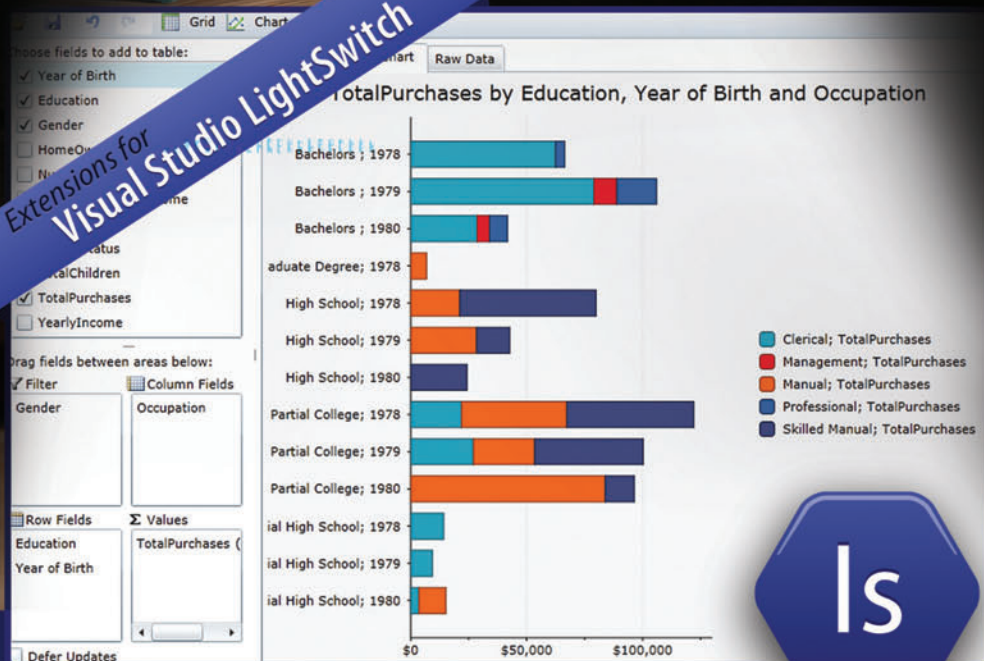
If you don’t want to create a custom instance, you can use the LocalDB built-in default instance named `v11.0`. To access the instance just register a connection to `“(localdb)\v11.0”` in Juneau, and the instance will be automatically created for you by LocalDB.

Because LocalDB is new, a patch to the Microsoft .NET Framework 4 is required to use the `(LocalDB)` prefix when accessing an instance through SSMS or managed application code. Installing Juneau will include this patch. Support for the prefix will be built into the next version of the .NET Framework.

Database developers face a problem similar to what Web developers faced (and which IISExpress solved for them): how to develop and debug code that requires a server to execute without needing to run a full server product locally on the development machine. LocalDB provides a solution for database developers. Because LocalDB is a crucial part of database development, it’s installed on your desktop when you install Juneau. When a database project is added to a solution, Juneau creates a new LocalDB instance named after the solution and creates a database within that instance for each database project. Juneau creates the data and log file for each database project in a directory within the project’s directory. For more information about LocalDB, see go.microsoft.com/fwlink/?LinkId=221201.

BUSINESS INTELLIGENCE *in the Flip of a Switch*

Extensions for
Visual Studio LightSwitch



componentone.com/lightswitch

 **ComponentOne**

Figure 6 Options for Controlling Updating Behavior

Behavior	Option	Result
Data loss	Block incremental deployment if data loss might occur	The incremental update engine will halt execution if a table has data and the script later makes a destructive change. A destructive change would include dropping a column or making a type change (bigint to int).
	Back up database before deployment	The engine will script a backup of the database prior to performing any changes.
Generating drops	Always recreate database	The update script will be written to detect if the database exists and, if it does, to set it into single-user mode and drop it.
	Drop objects in target but not in source	This option is a “hammer” that drops all objects in the target database that are not also in the source. This option overrides any data-loss options.
	Drop constraints, indexes, dml triggers	Treats constraints, indexes and dml triggers as if they’re part of the table or view; thus, removal of these objects from the project causes a drop of the corresponding object in the target.
	Drop permissions, extended properties	Similar to the previous option, treats these objects as if they’re part of their parent; thus, absence in the source causes target-only objects to be dropped.
Verifying new constraints	Check new constraints	When new foreign key and check constraints are created, they can “check” that existing data conforms to the constraint. The data check for new constraints is deferred to the end of the deployment script so that a constraint violation doesn’t cause an error in the middle of the deployment.
Transactions	Include transactional scripts	When this option is enabled, the engine attempts to wrap the generated script within a transaction. Because some objects can’t be managed within a transaction, it’s possible that a portion of the script will not be within a transaction.

to a database project’s build, Juneau leverages another new feature in SQL Server Denali called SQL Server T-SQL Compiler Services. This component provides complete verification of your source code without the need to execute that source against a live SQL Server. To see this in action, add the following code after the CreateCustomer procedure:

```
go
create table dbo.ExtendedVerificationDemo
(
    c1 int null,
    c2 as c2 + 5
)
```

When the project is built, you’ll see the following error in the error list and output window:

```
E:\projects\MsdnDemo\MsdnDemo\CreateCustomer.sql(12,1): Error:
SQL01759: Computed column 'c2' in table 'ExtendedVerificationDemo' is
not allowed to be used in another computed-column definition.
```

As you can see, this error message is exactly what you’d see reported by the SQL Server engine—in this case because a computed column can’t reference itself. As valuable as this feature is, there are situations where you might need to disable it, for example:

- The verification engine is built based on the SQL Server Denali engine and it enforces rules based on that engine. If

your project uses deprecated syntax that has been removed in SQL Server Denali, it will fail verification.

- The verification engine does not understand objects referenced by a fully qualified three- or four-part name. The engine will flag a warning or error based on the SQL Server deferred name resolution rules, which you can read more about at bit.ly/pDStXE.

If you run into this type of limitation you can turn off extended verification at the file or project level.

After all the build-time warnings and errors have been addressed, verify that your code actually works.

Verifying Your Source

Once you’ve built your project, you need to run (and possibly debug) your code before you check it into source code control. You have several options depending on where you are in the development cycle and what you want to do, as indicated in **Figure 4**.

Migrating Changes

When your code reaches a stable point, it’s time to migrate it to an instance to be tested and eventually used by your customers.

Juneau has an incremental update engine that will generate an update script based on the difference between your source and the target database. Although there’s one underlying engine, it’s exposed in three different ways within Juneau to support migration of your changes, as **Figure 5** describes.

To support these three different scenarios, the incremental update engine exposes a number of different options to control behavior. You can expect the defaults for these settings to change over time as the team tunes each scenario. For example, in the design-time case it might make sense for the engine to be more aggressive about ensuring the change is made, and to care less about

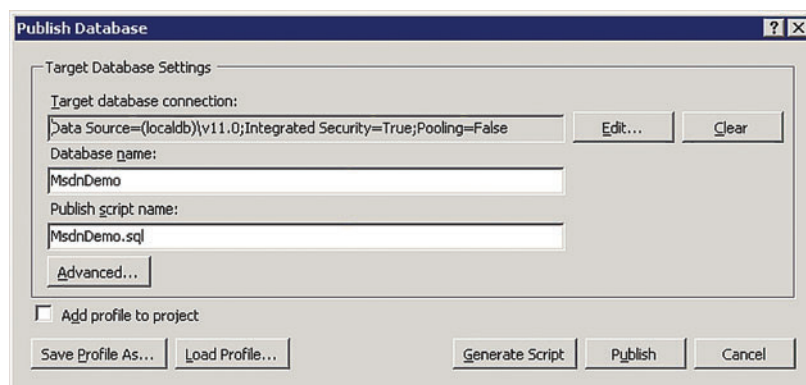


Figure 7 Publish Database Dialog

Powerful Tools for Developers

v4.5!



High-Performance PDF Printer Driver



- Create accurate PDF documents in a fraction of the time needed with other tools
- WHQL tested for all Windows 32 and 64-bit platforms
- Produce fully compliant PDF/A documents
- Standard PDF features included with a number of unique features
- Interface with any .NET or ActiveX programming language

v4.5!



PDF Editor for .NET, now Webform Enabled

- Edit, process and print PDF 1.7 documents programmatically
- Fast and lightweight 32 and 64-bit managed code assemblies for Windows, WPF and Web applications
- Support for dynamic objects such as edit-fields and sticky-notes
- Save image files directly to PDF, with optional OCR
- Multiple image compression formats such as PNG, JBIG2 and TIFF

New!



PDF Integration into Silverlight Applications

- Server-side PDF component based on the robust Amyuni PDF Creator ActiveX or .NET components
- Client-side C# Silverlight 3 control provided with source-code
- Optimization of PDF documents prior to converting them into XAML
- Conversion of PDF edit-boxes into Silverlight TextBox objects
- Support for other document formats such as TIFF and XPS



OCR Module available with royalty-free licensing!



The new OCR Module from Amyuni enables developers to:

- Convert non-searchable PDF files into searchable PDFs
- Create searchable PDF documents out of various image formats such as multi-page TIFF, JPEG or PNG while applying text recognition
- Compress image based PDF documents using high compression JBIG2 or more standard CCITT, JPEG and PNG compression formats

The Amyuni OCR module is based on the Tesseract Library with the Amyuni PDF technology being used to process and create the PDF documents.

Learn more at www.amyuni.com

More Development Tools Available at:

www.amyuni.com

USA and Canada
Toll Free: 1 866 926 9864
Support: (514) 868 9227
Info: sales@amyuni.com

Europe
Sales: (+33) 1 30 61 07 97
Support: (+33) 1 30 61 07 98
Customizations: management@amyuni.com

AMYUNI Technologies

All trademarks are property of their respective owners. © 1999-2010 AMYUNI Technologies. All rights reserved.

data loss because it's a debug database. Although there are many options, I'd like to call out a few behaviors and their corresponding options, as shown in **Figure 6**.

Being able to create an incremental update script is very helpful when migrating changes from a source to a target database. However, as precise as the incremental update script is, sometimes it's challenging to determine exactly what's occurring in the script or to summarize what the script is doing. As an aid to summarizing and understanding what the incremental script will perform, Juneau creates a preview report that highlights potential issues with the actions taken by the script, as well as summarizes the actions taken if the script is executed. For example, you can publish your project to the default LocalDB instance like this:

1. Open the background activity window by clicking on View | Other Windows | Database Activity Monitor.
2. Right-click on the project and select Publish.
3. Populate the Publish dialog, as shown in **Figure 7**.
4. Click Publish.

When publishing has completed, open the publish step and click on the View Plan link. A report like the one in **Figure 8** will be displayed.

As you can see, the report indicates that the two tables (we fixed the bug in the ExtendedVerification-Demo table) and procedure we wrote earlier will be created when the script is executed. The highlights section is empty, but you can see that the report will highlight actions that could cause a significant performance impact or data loss. If we had only generated a script rather than publishing, we could use this report to help verify the script prior to execution.

Connected Development

So far, we've talked about ways to use the Database Project to develop our code declaratively outside the context of a running database. All this technology is extremely useful when working in team environments, but sometimes you just want to interact with a live server! Microsoft knows that live servers are important to developers, and it has invested in a rich,

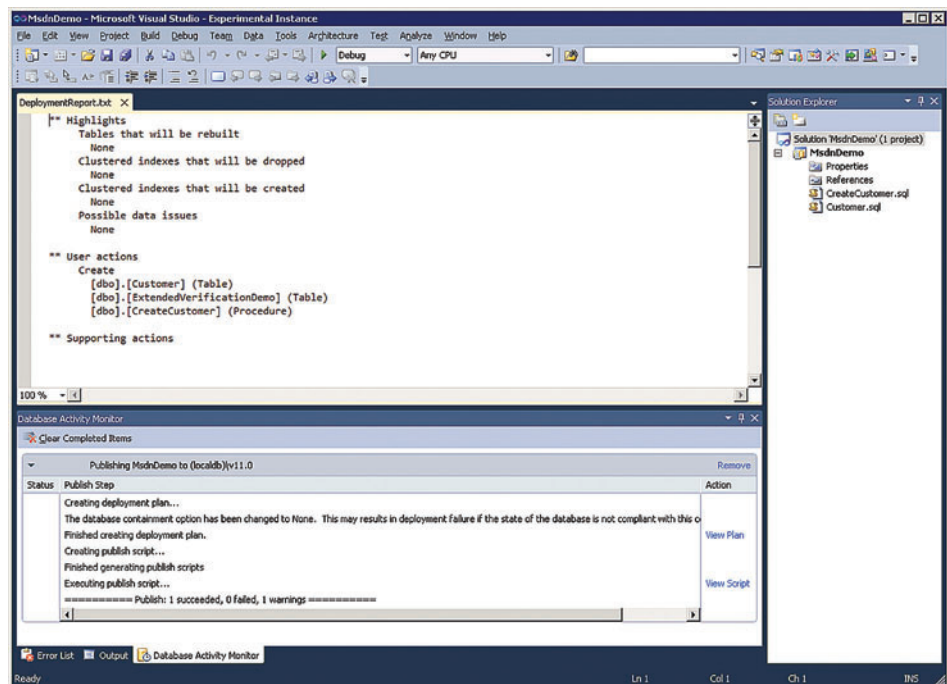


Figure 8 The Deployment Preview Report

developer-oriented, connected experience. To demonstrate, let's open the database to which we just published. To connect to that database:

1. Click on the View menu.
2. Select Server Explorer.
3. Right-click on the SQL Server node.
4. Select Add SQL Server.
5. Populate the connection dialog with (localdb)\v11.0.
6. Navigate to the just-published MsdnDemo database, as shown in **Figure 9**.

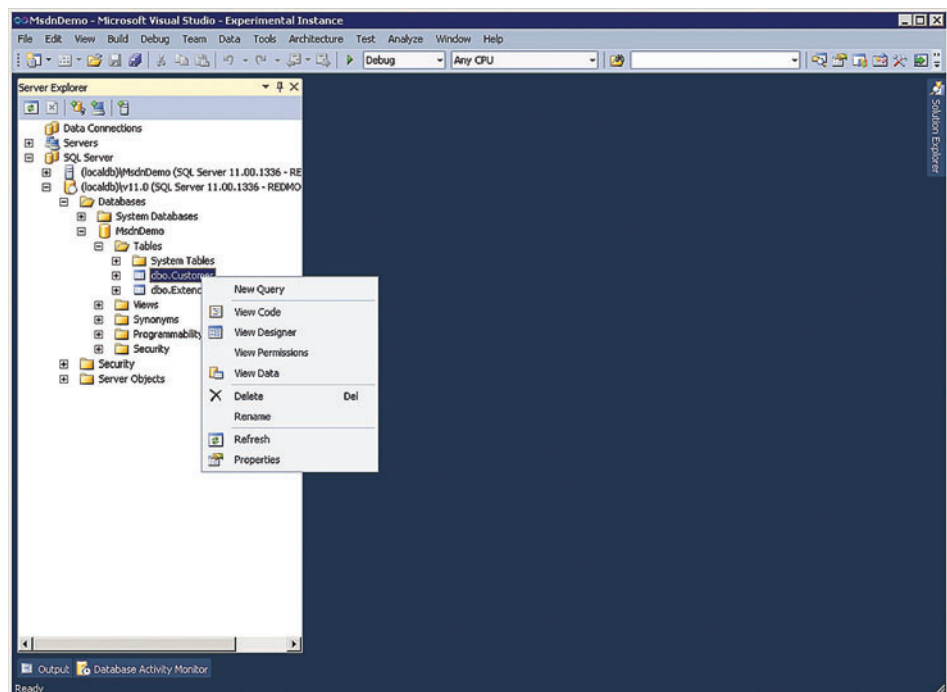


Figure 9 MsdnDemo Database in the New SQL Server Node in Server Explorer

Syncfusion: Facts in Black & White

We concentrate on stability and reliability. It is no secret. With a quiet commitment to customers and code, you may not be aware that we're the backbone of many applications. The reason: While others concentrate on flash-in-the-pan, we put sustenance on the table.



Time Tested

We've been with .NET since its beginning, and in that sense, we're proud to have some age on us. It means that our controls have been brought to maturity. Stability and reliability are their earmarks. When you drop one of our components in your application, you can be confident it's been tested not only by our QA, but also within the context of the real world.



Vast Offering

We've created one of the largest arrays of components on the market. Count the number of discrete controls, and you'll see. The breadth of our offering provides the flexibility to change all aspects of your application fully without relearning the differing minds of multiple vendors. With Syncfusion, all you need to know is one.



Forward Looking

With an eye on the horizon, we advance in HTML5, ASP.NET MVC, and Silverlight for web and Windows Phone 7. We possess a passion for the new. A thirst for progress. And we invite you to come see all are doing, and all we will do, in the new frontiers of development.

what we mean by 360° transformation

Syncfusion is your development partner—a helping hand outreached. In one ad, we couldn't possibly sell you on all our products; with one magazine, we couldn't possibly convince you of our commitment to service. But we can show you. We invite you to test our full 30-day evaluation—we are confident it is enough. During the trail, please utilize our support to see what we mean by 360° transformation.

Download at www.syncfusion.com/downloads/evaluation

 **Syncfusion®**
Deliver innovation with ease®

Figure 10 Update Database Preview Dialog

** Warnings	
	The object [dbo].[CreateCustomer] will be broken if the change is executed.
** Highlights	
Tables that will be rebuilt	None
Clustered indexes that will be dropped	None
Clustered indexes that will be created	None
Possible data issues	None
** User actions	
Drop	[dbo].[Customer] (Table)
** Supporting actions	

The first thing you might notice is that the tree looks very similar to the tree in SSMS. It should; the team wanted to limit the amount of relearning necessary when moving to Juneau. You can open a T-SQL Query editor right from the tree and begin writing code as you normally would; this editor is the same enhanced editor we talked about earlier. Unlike some of the SSMS capabilities, the Juneau tree is expressly meant for developer-oriented actions. For instance, if you right-click the table Customer and select Delete, you'll see a dialog that presents the same preview report you saw earlier. However, in this case there's a warning that the procedure dbo.CreateCustomer will be broken if the drop is executed, as shown in **Figure 10**.

A similar report is created if the table is renamed. In both these cases, before actions are executed, you're informed of the impact of the change; this report allows you to see what your changes will do to the database and what might need to be fixed (or deleted) should you apply the changes.

If you cancel the delete and then right-click on the Customer table and select View Designer, you'll see the same table designer you became familiar with in the project system—except this time it's hosted over the definition of the table retrieved from the server. As you might expect, the designer has a CREATE table statement using the same declarative programming model as the project system. In the designer, rename the Id column to CustomerId and add a second int column called Age. Now if you look in the error list, you'll see a warning that CreateCustomer has been broken by the column rename, as shown in **Figure 11**.

To fix this error you can either View Code on the CreateCustomer procedure or just double-click the warning and modify the insert statement to update the column names and supply @param2

as the value for the Age column. At this point, you have two windows (one source, one designer) with declarative object definitions retrieved from the database defining a set of changes to the objects on the server. If you click on the Update Database button, you'll see the now-familiar report that will tell you that the column will be renamed and both the table and procedure will be altered. Execute the script by clicking on Update Database and then navigate to the Customer table in Server Explorer. You'll see the tree update to include the CustomerId and Age columns.

The connected development experience in Server Explorer provides a great deal of power to you by supporting the same declarative programming model and enhancing online changes with real-time warning and error support as you make changes.

Getting the Bits

Juneau is available in the SQL Server Denali CTP3 release. It will also be available as a standalone Web download, and will integrate into existing installations of Visual Studio 2010 and the next version of Visual Studio. Moreover, Juneau has a separate command-line tools installer for publishing databases without requiring Visual Studio to be installed, and for Team Foundation Server automated-build scenarios. ■

JAMIE LAFLEN is a developer working on SQL Server Developer Tools. He previously worked on Visual Studio Database Projects.

BARCLAY HILL is a senior program manager working on SQL Server Developer Tools. He previously worked on Visual Studio Database Projects.

THANKS to the following technical experts for reviewing this article: Jeffrey Davis, Mike Kaufman, Dave Langer, Genevieve Orchard and Patrick Sirt

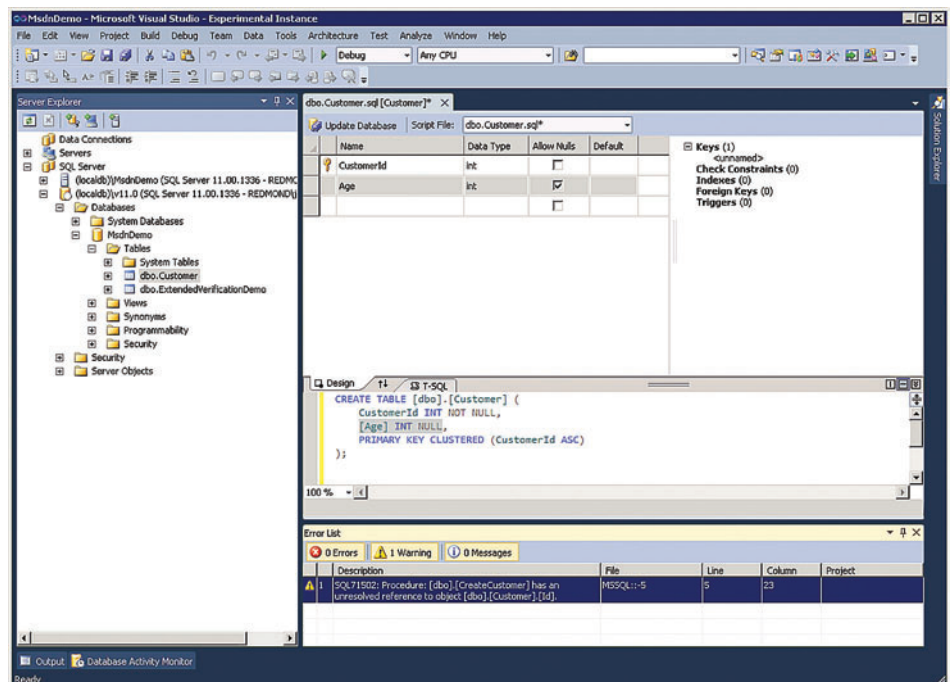


Figure 11 Error Resulting from Column Rename

CREATING EDITING PRINTING CONVERTING REPORTING ? IMPORTING or EXPORTING ?

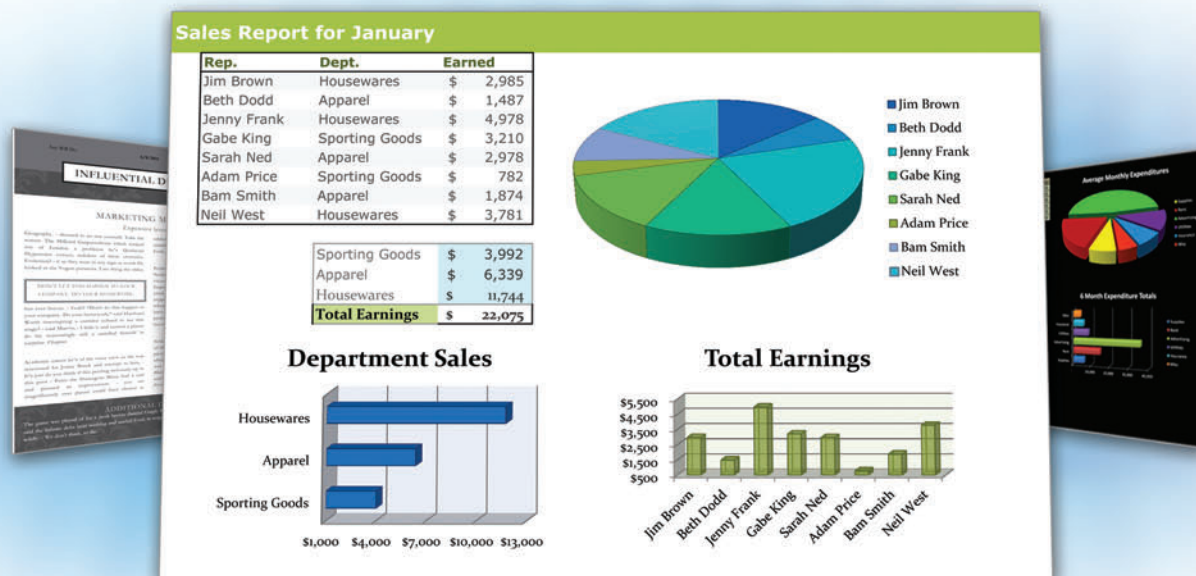
Documents Spreadsheets Presentations Barcodes Email

DOC DOCX	XLS HTML	PPT POTX	BMP JPG PNG	MSG
PDF HTML	TAB CSV	POT PPS	Supports 30+	MHT
TXT RTF	XLSX PDF	PPTX PDF	Symbologies	EML

...and many more file formats

100% Standalone - No Automation!

.Net Java Sharepoint SQL Reporting JasperReports



9,500+ Customers 37,500+ Licenses 130K+ User Community



Get your FREE evaluation copy at <http://www.aspose.com>.

US Sales: 1.888.277.6734 • sales@aspose.com • EU Sales: +44 (0)800 098 8425 • sales.europe@aspose.com
Enterprise Sales – enterprise.sales@aspose.com

Experience the Devexpress Difference

“As a training, mentoring and consulting company, we are often put on the spot as to which vendors we like for .Net tools. There is only one answer from my company and that is Developer Express. We have used Developer Express tools in our projects for the past five years and the company continues to impress me with the quality of their tools.

They simply work. You get what you pay for. Mileage will vary with other vendors but I can assure you Developer Express is a sure bet.”

—Mark Dunn, MCT, MCAD, MCDBA, MCSD .Net

“Our flagship product needed extensive visualizations including charts and graphs. We were looking for a single charting system that could address all of these needs, and handle large volumes of data. It needed to look attractive yet blend into our application.

With *XtraCharts* we were able to create high performance, real-time graphs of performance data through to detailed analytical bar charts. *XtraCharts* was the only option that was fast enough to handle the tens of thousands of data points our customers routinely throw at it.”

—Kendall Miller

Read More User Comments at:
DevExpress.com/Comments





#1 PRODUCT
ComponentSource Awards 2010-11



#1 PUBLISHER
ComponentSource Awards 2010-11



Award-Winning Presentation Controls and Reporting Libraries
For a **FREE** trial version visit us at **DevExpress.com/FreeEval**

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

DevExpress™

New Features in the Entity Framework June CTP

Srikanth Mandadi

The recently released Microsoft Entity Framework (EF) June 2011 CTP includes support for a number of frequently requested features, like enums, spatial types and table-valued functions (TVFs). We'll take a look at these features using simple walkthroughs. I'm assuming you're familiar with the EF (bit.ly/oLbjp0) and with the Code First development pattern (bit.ly/oQ77Hm) introduced in the EF 4.1 release.

Here's what you need to be able to try out the samples in this article:

- Visual Studio 2010 Express and SQL Server 2008 R2 Express or higher. You can download the Express editions of Visual Studio and SQL Server Express from bit.ly/rsFvxJ.
- The Microsoft EF and EF Tools June 2011 CTP. You can download these from bit.ly/mZgQIS.
- The Northwind database. You can download it from bit.ly/pwbDoQ.

Now, let's get started.

Enums

Let's start with one of the most requested features in the EF—enums. Many programming languages, including .NET languages like C# and Visual Basic, have support for enums. In the EF, the goal is to

allow users to have enums in their CLR types and map them to the underlying Entity Data Model (EDM), and in turn allow persisting these values to the database. Before going into detail, let's look at a simple example. Enums are supported in the Code First, Database First and Model First approaches. I'll start with the Database First approach and then show an example that uses the Code First approach.

For the Database First example, you'll use the Products table in the Northwind database. You need to make sure you're targeting the EF June 2011 CTP before adding the model, so be sure to:

1. Launch Visual Studio 2010 and create a new C# Console Application project.
2. Right-click on your project in Solution Explorer and select Properties.
3. Select Microsoft Entity Framework June 2011 CTP from the Target framework drop-down (see **Figure 1**).
4. Press Ctrl+S to save the project. Visual Studio asks for permission to close and reopen the project; click Yes.
5. Add a new model to the project by right-clicking Project | Add New Item (or Ctrl+Shift+A), and selecting ADO.NET Data Entity Model from Visual C# Items (we called ours "CTP1EnumsModel.edmx"), then click Add.
6. Follow the steps of the wizard to point to the Northwind database. Select the Products table and click Finish.
7. The Entity Model resulting from these choices has a single Entity, as shown in **Figure 2**.

Here's a LINQ query to get all products belonging to the Beverages category. Note that the CategoryID for Beverages is 1:

```
var ctx = new NorthwindEntities();  
var beverageProducts = from p in ctx.Products  
                        where p.CategoryID == 1
```

This article discusses:

- New features—enums, table-valued functions and spatial types
- Using the Code First and Database First approaches
- Auto-compiled LINQ queries

Technologies discussed:

Entity Framework 4.2, Visual Studio 2010

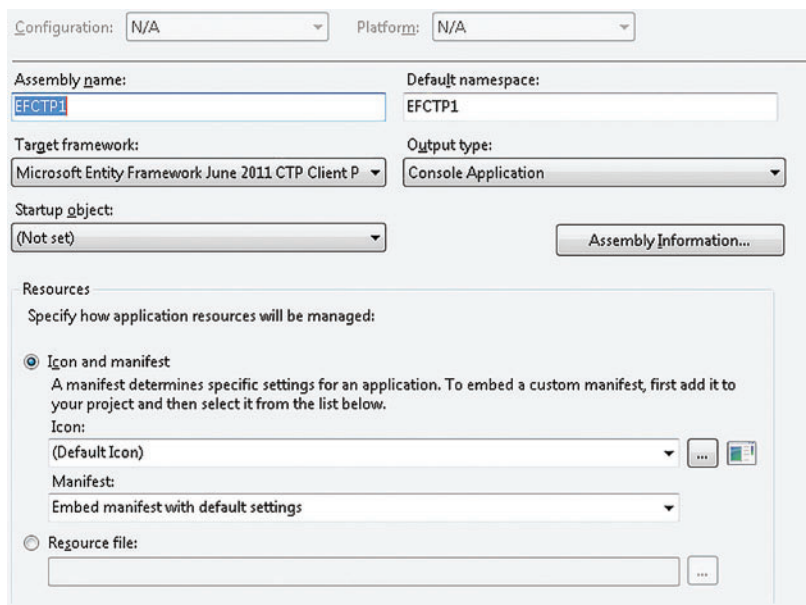


Figure 1 Targeting the Entity Framework June 2011 CTP

Of course, to write this query you'd have to know that CategoryID for Beverages is 1, which would only be possible if you remembered it or went to the database to look up the value. The other problem with this query is that it's not clear which Category the code is querying for. It needs either some documentation in every place CategoryID is used, or some knowledge on the part of the person reading the code about the CategoryID values for various Categories.

Another problem pops up when you try to insert a new Product. Use the following code to do an insert into the Products table:

```
var ctx = new NorthwindEntities();
var product = new Product() { ProductName = "place holder",
    Discontinued = false, CategoryID = 13 };
ctx.AddToProducts(product);
ctx.SaveChanges();
```

When you run this code, you'll get a foreign key constraint exception from the database. The exception is thrown because there's no Category with an ID value of 13. It would be much nicer if the programmer had a way to know the list of categories and assign the correct one instead of having to remember the set of valid integer values.

Let's introduce enums into the model and see how this improves the scenarios.

Here are the steps to change CategoryID into an enum:

1. Open the model in the designer by double-clicking on the CTPIEnumsModel.edmx file.
2. Right-click CategoryID property in Product Entity and choose Convert to Enum.
3. Create an enum type and enter the values for enums members in the new dialog that opens up (see Figure 3). Name the enum type Category and select the underlying type as Byte. The underlying type is the integral type that represents the value

space for enums. You can choose it based on the number of members in the enum. For this particular enum, there are eight members, which fit in a byte. Enter the members in the ascending order of the CategoryID values. Enter the Value for the first Category (Beverages) as 1 and leave the Value field empty for other members because the values for the other members automatically increment by 1 in the database. This is the default chosen by the EF as well. But if the values were different in the database, you'd enter them in the Value field for all the Categories. If the value of Beverages was 0 instead of 1, you could leave that empty as well because the EF chooses 0 as the default for the first member of an enum.

4. When creating an enum, you can choose to designate it as a flag using the "Is Flag?" option. This is only used during code generation; if it's checked, the enum type will be generated with a Flags attribute (see bit.ly/oPqiMp for more information on flags enumerations). For this example, leave the option unchecked.

5. Rebuild the application to regenerate the code, and the resulting code now includes enums.

You can rewrite the query to get all products that are Beverages as follows:

```
var ctx = new NorthwindEntities();
var beverageProducts = from p in ctx.Products
    where p.Category == Category.Beverages
    select p;
```

Now IntelliSense helps you write the query, rather than you having to go to the database to find the value of Beverages. Similarly, in updates, IntelliSense will show me the correct values for the Category.

We just looked at enums using a Database First approach. Now I'll use the Code First approach to write the query to get all Beverages using enums. To do this, create another Console Application and add a C# file with the types shown in Figure 4.

The class EnumsCodeFirstContext inherits from DbContext. DbContext is a new type that shipped in the EF 4.1 and is similar toObjectContext—but is much simpler and more straightforward to use. (For more information on using the DbContext API, see bit.ly/eeEsyt.)

A couple of things to note in the code in Figure 4:

- The Column attribute above the Category property: This is used to map between CLR properties and columns when the two have different names or types.
- The constructor for EnumsCodeFirstContext, which invokes the base class constructor by passing in a connection string: By default, DbContext creates a database in the local SqlExpress with a fully qualified class name that derives from DbContext. For this sample, we simply use the existing Northwind database.

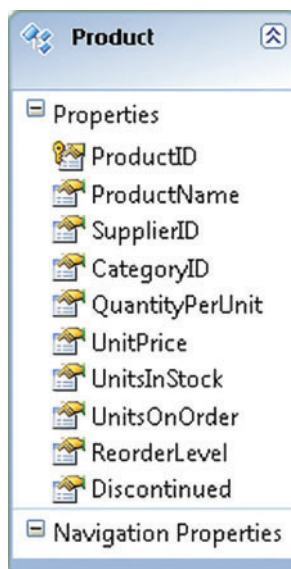


Figure 2 Entity Model for Product Entity

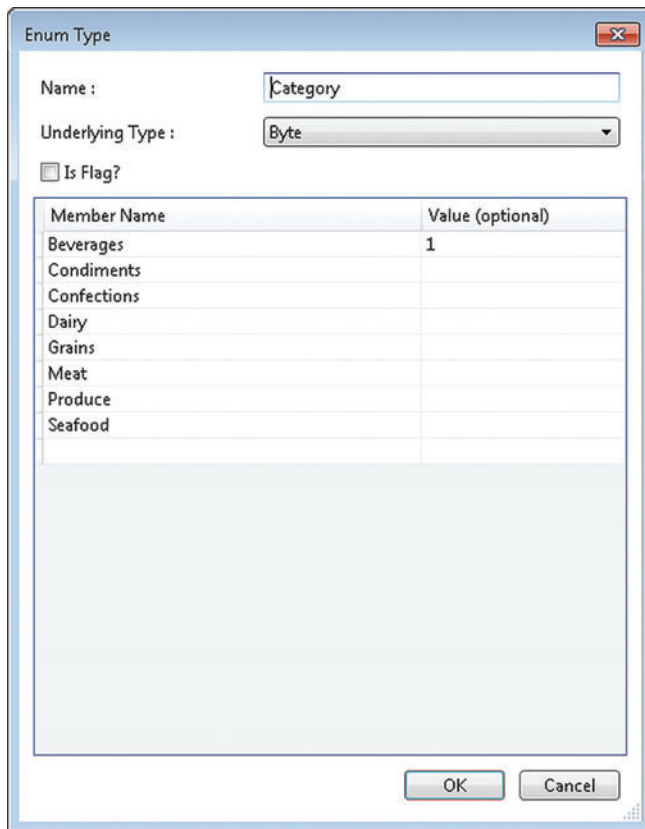


Figure 3 The Enum Type Creation Window

Now you can write code similar to the Database First context to get all the products that belong to the Beverages Category:

```
EnumsCodeFirstContext ctx = new EnumsCodeFirstContext();
var beverageProducts = from p in ctx.Products
    where p.Category == Category.Beverages
    select p;
```

Another significant feature added in this CTP is support for table-valued functions (TVFs).

Table-Valued Functions

Another significant feature added in this CTP is support for TVFs. TVFs are very similar to stored procedures with one key difference: the result of a TVF is composable. That means the results from a TVF can be used in an outer query. So, a major implication for developers using the EF is that a TVF can be used in a LINQ query while a stored procedure can't. I'll walk through an example that shows how TVFs can be used in an EF application. In the process, you'll see how to take advantage of the Full-Text Search (FTS) functionality in SQL Server (see bit.ly/qZXG9X for more information).

FTS functionality is exposed via a couple of predicates and TVFs. In previous versions of the EF, the only way to use the full-text TVFs would be to either invoke them in a T-SQL script using

ExecuteStoreCommand or use a stored procedure. But both of these mechanisms are not composable and can't be used in LINQ to Entities. My example will show you how to use these functions as composable functions with the support for TVFs in this CTP. To do this, I've taken a query from MSDN documentation for ContainsTable (bit.ly/q8FFws). The query searches for all product names containing the words "breads," "fish" or "beers," and different weights are given to each word. For each returned row matching these search criteria, the relative closeness (ranking value) of the match is shown:

```
SELECT FT_TBL.CategoryName, FT_TBL.Description, KEY_TBL.RANK
FROM Categories AS FT_TBL
INNER JOIN CONTAINSTABLE(Categories, Description,
    'ISABOUT (breads weight (.8),
    fish weight (.4), beers weight (.2))' ) AS KEY_TBL
ON FT_TBL.CategoryID = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC;
```

Let's try to write the same query in LINQ to Entities. Unfortunately, you can't expose ContainsTable directly to the EF because it expects the first two parameters (table name and column name) as unquoted identifiers—that is, as Categories instead of 'Categories,' and there's no way to tell the EF to treat these parameters specially. To work around this limitation, wrap ContainsTable in another user-defined TVF. Execute the following SQL to create a TVF called ContainsTableWrapper (the TVF executes the ContainsTable function on the Description column in Categories table):

```
Use Northwind;
Create Function ContainsTableWrapper(@searchstring nvarchar(4000))
returns table
as
return (select [rank], [key] from ContainsTable(Categories, Description,
    @searchstring))
```

Now, create an EF application and use this TVF. Follow the same steps as described for the enums example to create a console application and add an entity model by pointing to Northwind. Include Categories, Products and the newly created TVF. The model will look like the one shown in Figure 5.

The TVF is not shown on the designer surface, but you can view it in the Model Browser by expanding the Stored Procedures/ Functions in the Store section.

To use this function in LINQ, add a function stub (as described at bit.ly/qhlye2). I added the function stub in a partial class for theObjectContext class, in this case NorthwindEntities:

```
public partial class NorthwindEntities
{
    [EdmFunction("NorthwindModel.Store", "ContainsTableWrapper")]
    public IQueryable<DbDataRecord> ContainsTableWrapper(string searchString)
    {
        return this.CreateQuery<DbDataRecord>(
            "[NorthwindModel.Store].[ContainsTableWrapper](@searchstring)",
            new ObjectParameter[] {
                new ObjectParameter("searchString", searchString));
    }
}
```

You can now start using this function in your queries. Simply print out the Key—that is, CategoryId and Rank for the full-text query, mentioned earlier:

```
var ctx = new NorthwindEntities();
var fulltextResults = from r in ctx.ContainsTableWrapper("ISABOUT
    (breads weight (.8),
    fish weight (.4), beers weight (.2))")
    select r;
foreach (var result in fulltextResults)
{
    Console.WriteLine("Category ID: " + result["Key"] + " Rank : " + result["Rank"]);
}
```

"T" is for Tomorrow.

// Build/ the future with Telerik.

The leading developer tools for web, desktop, and mobile. Today and Tomorrow.

www.telerik.com/tomorrow

 **telerik**
deliver more than expected

Figure 4 Using Enums with a Code First Approach

```
public enum Category : byte
{
    Beverages = 1,
    Condiments,
    Confections,
    Dairy,
    Grains,
    Meat,
    Produce,
    Seafood
}

public class Product
{
    public int ProductID { get; set; }
    public string ProductName { get; set; }
    public int? SupplierID { get; set; }
    [Column("CategoryID", TypeName = "int")]
    public Category Category { get; set; }
    public string QuantityPerUnit { get; set; }
    public decimal? UnitPrice { get; set; }
    public short? UnitsInStock { get; set; }
    public short? UnitsOnOrder { get; set; }
    public short? ReorderLevel { get; set; }
    public bool Discontinued { get; set; }
}

public class EnumsCodeFirstContext : DbContext
{
    public EnumsCodeFirstContext() : base(
        "data source=<server name>; initial catalog=Northwind;
        integrated security=True;multipleactiveresultsets=True;"
    )
    {
    }
    public DbSet<Product> Products { get; set; }
}
```

The output on the console when you run this piece of code is as follows:

```
Category ID:1   Rank :15
Category ID:3   Rank :47
Category ID:5   Rank :47
Category ID:8   Rank :31
```

But this isn't the query we were trying to write. The one we want actually does a little more. It provides the Category Name and Description along with the rank, which is more interesting than just the Category ID. Here's the original query:

```
SELECT FT_TBL.CategoryName, FT_TBL.Description, KEY_TBL.RANK
FROM Categories AS FT_TBL
INNER JOIN CONTAINSTABLE(Categories, Description,
    'ISABOUT (breads weight (.8),
    fish weight (.4), beers weight (.2) )' ) AS KEY_TBL
ON FT_TBL.CategoryID = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC;
```

In order to use LINQ for this query, you need to map the TVF to a Function Import in EDM with a Complex type or an Entity return type, because Function Imports that return Row Types are not composable.

To do the mapping, here's what you need to do:

1. Double-click the Store function in the model browser to open the Add Function Import dialog shown in **Figure 6**.
2. Enter the Function Import Name as ContainsTableWrapperModelFunction.
3. Check the Function Import is Composable? box.
4. Select the ContainsTableWrapper function from the drop-down list for Stored Procedure/Function Name.
5. Click the GetColumnInformation button to populate the

table below the button with information about the result type returned from the function.

6. Click the Create New Complex Type button. It results in changing the selection of "Returns a Collection Of" radio button to Complex with a generated name for the complex type.
7. Click OK.

For the TVFs that are mapped to Function Imports in the model, you don't need to add a corresponding function in the code because the function is generated for you.

Now you can write the T-SQL query that was using the FTS function ContainsTable in LINQ, as follows:

```
var ctx = new NorthwindEntities();
var fulltextResults = from r in ctx.ContainsTableWrapperModelFunction("ISABOUT
    (breads weight (.8), fish weight (.4), beers weight (.2) )"
    join c in ctx.Categories
    on r.key equals c.CategoryID
    select new { c.CategoryName, c.Description, Rank = r.rank };

foreach (var result in fulltextResults)
{
    Console.WriteLine("Category Name:" + result.CategoryName + " Description:" +
        result.Description + " Rank:" + result.Rank);
}
```

When you run this code, the output on the console is:

```
Category Name:Beverages Description:Soft drinks, coffees, teas, beers,
and ales Rank:15
Category Name:Confections Description:Desserts, candies, and sweet
breads Rank:47
Category Name:Grains/Cereals Description:Breads, crackers, pasta, and
cereal Rank:47
Category Name:Seafood Description:Seaweed and fish Rank:31
```

Spatial Types Support

Now let's look at a new feature that has caused a lot of excitement: support for spatial types. Two new types have been added to the EDM—DbGeometry and DbGeography. The next example will show you how straightforward it is to use spatial types in the EF using Code First.

Create a ConsoleApplication project called EFSpatialSample, then add a C# file to this project with the following types:

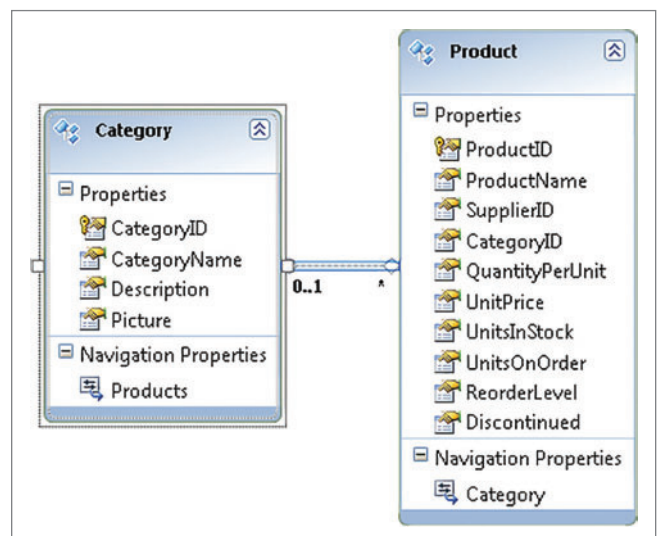


Figure 5 Entity Model with Products and Categories from Northwind

Our Name Says It All

activePDF®

Come and see why thousands of customers have trusted us over the last decade for all their server based PDF development needs.

- . Convert over 400 files types to PDF
- . High fidelity translation from PDF to Office
- . ISO 32000 Support including PDF/X & PDF/A
- . HTML5 to PDF
- . True PDF Print Server
- . Form Fill PDF
- . Append, Stamp, Secure and Digitally Sign



Download our FREE evaluation from
www.activepdf.com/MSDN

Call 1-866-GoTo-PDF | 949-582-9002 | Sales@activepdf.com

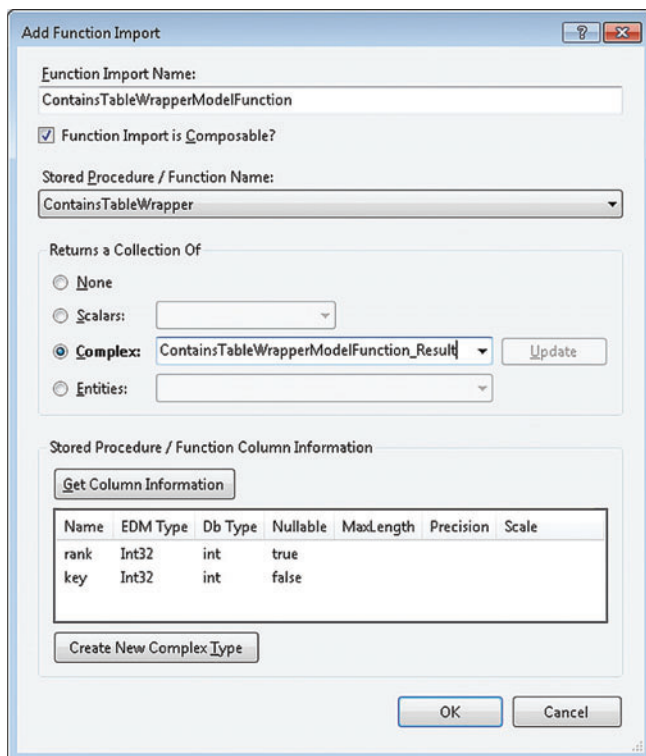


Figure 6 Mapping a TVF to a Function Import

```
namespace EFCTPSpatial
{
    public class Customer
    {
        public int CustomerID { get; set; }
        public string Name { get; set; }
        public DbGeography Location { get; set; }
    }

    public class SpatialExampleContext : DbContext
    {
        public DbSet<Customer> People { get; set; }
    }
}
```

The Location property in Customer is of type DbGeography, which has been added to the System.Data.Spatial namespace in this CTP. DbGeography is mapped to SqlGeography in the case of SQL Server. Insert some spatial data using these types and then use LINQ to query for the data (see Figure 7).

Figure 7 Working with Spatial Data

```
static void Main(string[] args)
{
    var ctx = new SpatialExampleContext();
    ctx.Customers.Add(new Customer() { CustomerID = 1, Name = "Customer1",
        Location = DbGeography.Parse(("POINT(-122.336106 47.605049)")) });
    ctx.Customers.Add(new Customer() { CustomerID = 2, Name = "Customer2",
        Location = DbGeography.Parse(("POINT(-122.31946 47.625112)")) });
    ctx.SaveChanges();

    var customer1 = ctx.Customers.Find(1);
    var distances = from c in ctx.Customers
        select new { Name = c.Name, DistanceFromCustomer1 =
            c.Location.Distance(customer1.Location) };
    foreach (var item in distances)
    {
        Console.WriteLine("Customer Name:" + item.Name + ",
            Distance from Customer 1:" + (item.DistanceFromCustomer1 / 1609.344 ));
    }
}
```

What happens in the code is pretty simple. You create two Customers with two different locations and specify their locations using Well-Known Text (you can read more about this protocol at bit.ly/owlhfu). These changes are persisted to the database. Next, the LINQ to Entities query gets the distance of each customer in the table from Customer1. The distance is divided by 1609.344, which converts from meters to miles. Here's the output from the program:

```
Customer Name:Customer1, Distance from Customer 1:0
Customer Name:Customer2, Distance from Customer 1:1.58929160985881
```

As expected, the distance from Customer1 to Customer 1 is zero. The distance of Customer 1 from Customer 2 is in miles. The Distance operation in the query gets executed in the database using the STDistance function. Here's the SQL query that gets sent to the database:

```
SELECT 1 AS [C1], [Extent1].[Name] AS [Name], [Extent1].[Location].
STDistance(@p__linq__0)
AS [C2]
FROM [dbo].[Customers] AS [Extent1]
```

Compiling the expression tree into SQL involves some overhead.

Auto-Compiled LINQ Queries

When you write a LINQ to Entities query today, the EF walks over the *expression* tree generated by the C# or Visual Basic compiler and translates (or compiles) it into SQL. Compiling the expression tree into SQL involves some overhead, though, particularly for more complex queries. To avoid having to pay this performance penalty every time the LINQ query is executed, you can compile your queries and then reuse them. The CompiledQuery class allows you to pay the overhead of compilation just once, and gives you back a delegate that points directly at the compiled version of the query in the EF cache.

The June CTP supports a new feature called the Auto-Compiled LINQ Queries, which lets every LINQ to Entities query you execute automatically get compiled and placed in the EF query cache. Every time you run the query subsequently, the EF will find it in its query cache and won't have to go through the whole compilation process again. This feature also provides a boost to queries issued using WCF Data Services, as it uses LINQ under the covers. For more on expression trees, see bit.ly/o5X3rA.

Wrapping Up

As you can see, there are a number of exciting features coming in the next release of the EF—and there are even more than those I've been able to touch on here, such as SQL generation improvements for Table per Type, or TPT, mapping and multiple result sets from stored procedures. The CTP gives you a chance to try out the bits and provide feedback to fix any bugs or improve the experience of the new features. You can report any bugs using the Microsoft Data Developer Connect site (connect.microsoft.com/data), or suggest new features via the EF user voice (ef.mswish.net) Web site. ■

SRIKANTH MANDADI is a development lead on the Entity Framework team.

THANKS to the following technical experts for reviewing this article:
The Entity Framework Team



BEST SELLER

TX Text Control .NET for Windows Forms/WPF

from \$499.59



Word processing components for Visual Studio .NET.

- Add professional word processing to your applications
- Royalty-free Windows Forms and WPF rich text box
- True WYSIWYG, nested tables, text frames, headers and footers, images, bullets, structured numbered lists, zoom, dialog boxes, section breaks, page columns
- Load, save and edit DOCX, DOC, PDF, PDF/A, RTF, HTML, TXT and XML



BEST SELLER

FusionCharts

from \$195.02



Interactive Flash & JavaScript (HTML5) charts for web apps.

- Liven up your web applications using animated & data-driven charts
- Create AJAX-enabled charts with drill-down capabilities in minutes
- Export charts as images/PDF and data as CSV from charts itself
- Create gauges, dashboards, financial charts and over 550 maps
- Trusted by over 18,000 customers and 375,000 users in 110 countries



BEST SELLER

ActiveAnalysis

from \$979.02



Rapidly embed out-of-the box OLAP, data visualization and BI features to your applications.

- Charts, pivot tables and data visualization in one programmable control
- Support for Silverlight, Windows Forms and ASP.NET development in one component
- Rich drag-and-drop user experience encourages self-discovery of data
- Excel exports allow end users to share analysis results offline
- Flexible data binding allows you to treat relational data as multi-dimensional



NEW RELEASE

Janus WinForms Controls Suite V4.0

from \$853.44



Add powerful Outlook style interfaces to your .NET applications.

- Includes Ribbon, Grid, Calendar view, Timeline and Shortcut bar
- Now features Office 2010 visual style for all controls
- Visual Studio 2010 and .NET Framework Client Profiles support
- Janus Ribbon adds Backstage Menus and Tab functionality as in Office 2010 applications
- Now features support for multiple cell selection

Build Great Experiences on Any Device with OData

Shayne Burgess

In this article, I'll show you what I believe is a serious data challenge facing many organizations today, and then I'll show you how the Open Data Protocol (OData) and its ecosystem can help to alleviate that challenge. I'll then go further and show how OData can be used to build a great experience on Windows Phone 7 using the new OData library for Windows Phone 7.1 beta (code-named "Mango").

The Data Reach Challenge

An interesting thing happened at the end of 2010: for the first time in history, the number of smartphone shipments surpassed the number of PC shipments. A key thing to note is that this isn't a one-horse race. There are many players (Apple Inc., Google Inc., Microsoft, Research In Motion Ltd. and so on) and platforms

(desktop, Web, phones, tablets and more coming all the time). Many organizations want to target client experiences on all or most of these devices. And many organizations also have a variety of data and services that they want to make available to client devices. These may be on-premises, available through traditional Web services or built in the cloud.

The combination of a large and ever-expanding variety of client platforms with a large and ever-expanding variety of services creates a significant cost and complexity problem. When support for a new client platform is added, the data and services often have to be updated and modified to support that platform. And when a new service is added, all of the existing client platforms need to be modified to support that service. This is what I consider to be the data reach problem. How can we define a service that's flexible enough to suit the needs of all the existing client experiences—and new client experiences that haven't been invented yet? How can we define client libraries and applications that can work with a variety of services and data sources? These are some of the key questions that I hope to show can be partly addressed with OData.

The Open Data Protocol

OData is a Web protocol for querying and updating data that provides a uniform way to unlock your data. Simply put, OData provides a standard format for transferring data and a uniform interface for accessing that data. It's based on ATOM and JSON feeds and the interface uses standard HTTP (REST) interfaces for exposing querying and updating capabilities. OData is a key component in bridging the data reach problem because, being a uniform, flexible interface, the API can be created once and used from a variety of client experiences.

This article discusses a prerelease version of Windows Phone 7.1 (code-named "Mango"). All information is subject to change.

This article discusses:

- The challenge of targeting multiple platforms and devices
- OData and its ecosystem
- The Windows Phone 7 OData Library
- Supporting LINQ
- Tombstoning in Windows Phone 7
- Providing credentials to an OData service

Technologies discussed:

Open Data Protocol, Windows Phone 7, Microsoft .NET Framework, iOS, LINQ

Figure 1 A Sampling of the OData Ecosystem

Clients
.NET
Silverlight
Windows Phone
WebOS
iPhone
DataJS
Excel
Tableau
Services
Windows Azure
SQL Azure
DataMarket
SharePoint
SAP
Netflix
eBay
Facebook
Wine.com

The OData Ecosystem

The flexible OData interface that can be used from a variety of clients is most powerful when there are a variety of client experiences that can be built using existing OData-enabled clients. The OData client ecosystem has evolved over the last few years to the point where there are client libraries for the majority of client devices and platforms, with more coming all the time. **Figure 1** lists just a sampling of the client and server platforms available at the time this article was written (a much more complete list of existing OData services is at odata.org/producers).

I can generally target any of the listed services in **Figure 1** with any client. To help demonstrate this in practice, I'll give examples of OData flexibility using a couple of different clients.

I'll use the Northwind OData (read-only) sample service available on odata.org at <http://services.odata.org/Northwind/Northwind.svc/>. If you're familiar with the

Northwind database, the Northwind service shown in **Figure 2** simply exposes the Northwind model as an OData service directly. **Figure 2** shows the service document in the Northwind service from the browser; the service document exposes the entity sets in the service that are basically just the tables from the Northwind database.

Let's look at a couple of examples of using the client libraries to consume and build experiences using the Northwind OData service. For the first example, let's consider a client experience that doesn't require any code to build. The PowerPivot add-in for Excel provides an in-memory analysis engine that can be used directly from within the Excel interface and supports importing data directly from an OData feed. The PowerPivot add-in also includes support for importing data directly from the Windows Azure Marketplace DataMarket, as well as a number of other data sources.

Figure 3 shows a pivot chart created by importing the Products and Orders data from the Northwind OData service and joining the result. The price-per-unit and quantity fields of the order details are combined to produce a total value for each order, and then the average total value for each order is shown, grouped by product and

then displayed on a simple pivot chart for analysis. All of this is done using the built-in interface and tools in PowerPivot and Excel.

Let's look next at another example of how to build an OData client experience on a different platform: iOS. The OData client library for iOS lets you build apps on that platform (such as for the iPad and iPhone) that consume existing OData services. The iOS library for OData includes a code-generation tool, *odatagen*, which will generate a set of proxy classes from the metadata on the

The OData client library for iOS lets you build apps on that platform (such as for the iPad and iPhone) that consume existing OData services.

service. The proxy classes provide facilities for generating OData URIs, deserializing a response into client-side objects and issuing create, read, update and delete (CRUD) operations against the service. The following code snippet shows an example of executing a request for the set of customers from the OData service:

```
// Create the client side proxy and specify the service URL.
NorthwindCatalog *proxy =
[[NorthwindCatalog alloc] initWithUri:@"http://host/Northwind.svc"];

// Create a query.
DataServiceQuery *query = [proxy Customers];

QueryOperationResponse *response = [query execute];
customers = [query getResults];
```

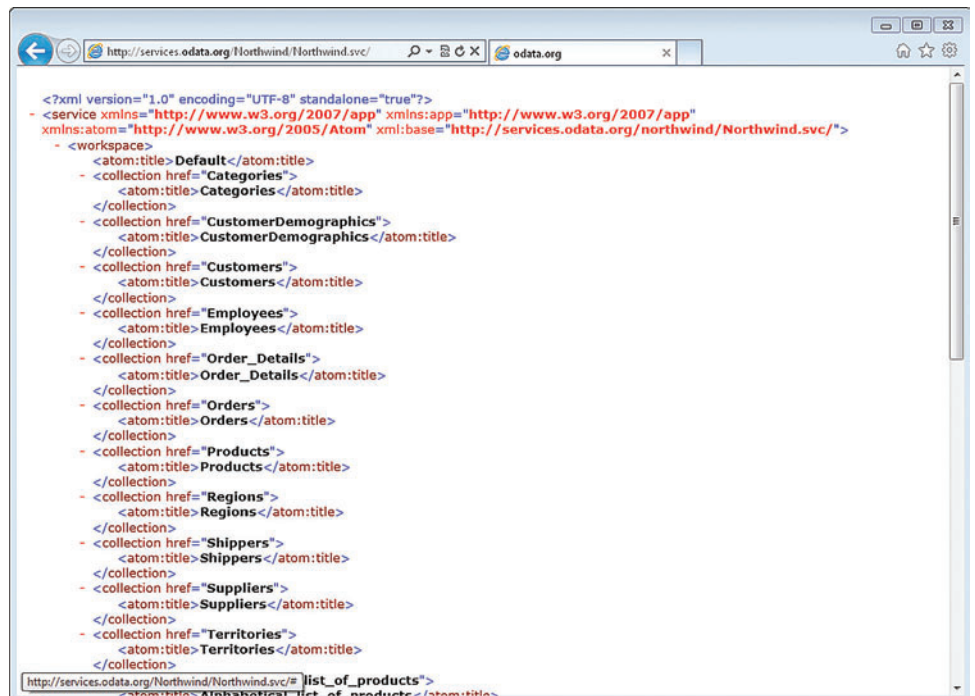


Figure 2 The Northwind Service

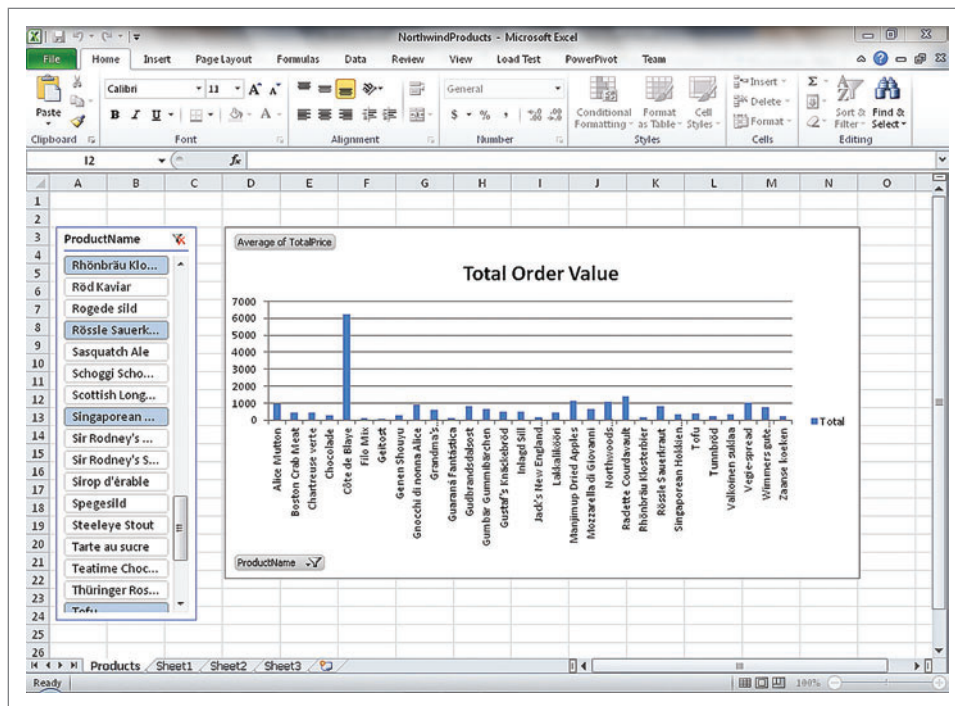


Figure 3 PowerPivot Add-In

The DataServiceQuery and QueryOperationResponse are used to execute a query by generating a URI and executing it against the service.

The Windows Phone 7 Library

Next, I'll demonstrate another example of an OData library on a third platform. I did a quick overview of building on OData on other platforms, but for this one I'll do a detailed walk-through to give insight into what it takes to build an app. To do this, I'll show you the basic steps needed to write a data-driven experience on Windows Phone 7 using the new Windows Phone 7.1 (Mango) tools. For the rest of the article, I'll walk through the key considerations when building a Windows Phone 7 app and show how the OData library is used. The app I build will target the same Northwind sample I used in the previous two examples (the Excel PowerPivot add-in and the iOS library).

The latest release of the Windows Phone 7.1 (Mango) beta tools features a significant upgrade in the SDK support for OData. The new library supports a set of new features that makes creating OData-enabled Windows Phone 7 apps easier. The Visual Studio Tools for Windows Phone 7 were updated to support the generation of a custom client proxy based on a target OData service. For those familiar with the existing Microsoft .NET Framework and Silverlight OData clients, this, too, will be familiar. To use the Visual Studio tools to automatically

generate the proxy in a Windows Phone 7 project, right-click the Service References node in the project tree and select Add Service Reference (note that this is only available if the new Mango tools are installed). Once the Add Service Reference dialog appears, enter the URI of the OData service you're targeting and select Go. The dialog will display the entity sets exposed by the service; you can then enter a namespace for the service and select OK. The first time the code generation is performed using Add Service Reference, it will use default options, but it's possible to further configure the code generation by clicking the Show All Files option in the solution explorer, opening the .datasvcmap file in the service reference and configuring the parameters. **Figure 4** shows the completed Add Service

Reference dialog with the list of entity sets available.

A command-line tool is also included in the Visual Studio Tools for Windows Phone 7 that's able to perform the same code-generation step. In either case, the code generation will create a client proxy class (DataServiceContext) for interacting with the service, and a set of proxy classes that reflect the shape of the service.

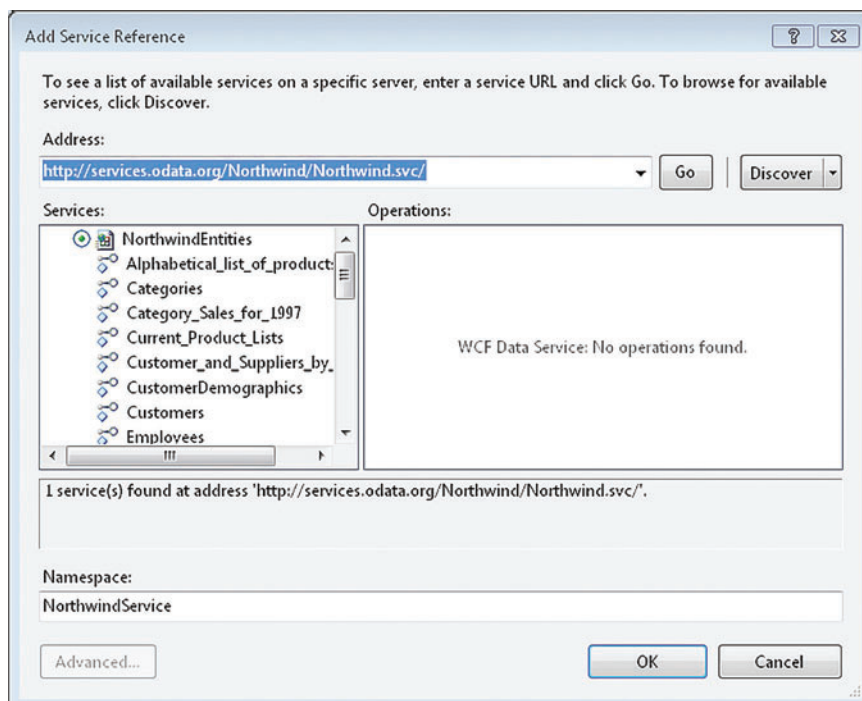


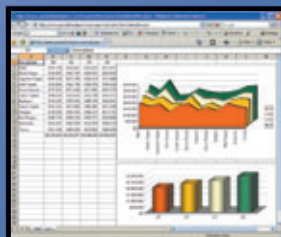
Figure 4 Using the Add Service Reference Dialog

20 Minutes to 4 Seconds...

SpreadsheetGear for .NET reduced the time to generate a critical Excel Report "from 20 minutes to 4 seconds" making his team "look like miracle workers."

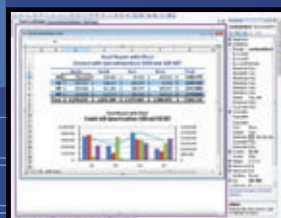
Luke Melia, Software Development Manager at Oxygen Media in New York

SpreadsheetGear 2010



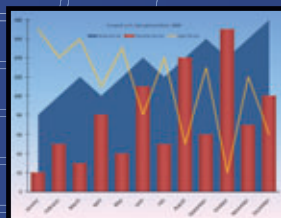
ASP.NET Excel Reporting

Easily create richly formatted Excel reports without Excel using the new generation of spreadsheet technology built from the ground up for scalability and reliability.



Excel Compatible Windows Forms Control

Add powerful Excel compatible viewing, editing, formatting, calculating, charting and printing to your Windows Forms applications with the easy to use WorkbookView control.



Create Dashboards from Excel Charts and Ranges

You and your users can design dashboards, reports, charts, and models in Excel rather than hard to learn developer tools and you can easily deploy them with one line of code.

Download the FREE fully functional 30-Day evaluation of SpreadsheetGear 2010 today at

www.SpreadsheetGear.com.



Toll Free USA (888) 774-3273 | Phone (913) 390-4797 | sales@spreadsheetgear.com

Figure 5 Executing a LINQ Query

```
public void LoadData()
{
    // Create a Northwind Entities context and set the URL of the service.
    NorthwindEntities svcContext = new NorthwindEntities(
        new Uri("http://services.odata.org/Northwind/Northwind.svc")
    );

    // Create a DataServiceCollection to hold the results of the query.
    // This collection implements INotifyCollectionChanged and can be bound in XAML.
    DataServiceCollection<Product> ProductsCollection =
        new DataServiceCollection<Product>( svcContext );

    // Execute a LINQ query for all products with some units in stock and
    // include the supplier.
    var q = from p in svcContext.Products.Expand("Supplier")
            where p.UnitsInStock > 0
            select p;

    // Asynchronously execute the query and load the results.
    ProductsCollection.LoadAsync(q);
}
```

Figure 6 XAML Binding

```
<ListBox x:Name="ProductListBox" Margin="0,0,-12,0"
ItemsSource="{Binding Products}">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <StackPanel Margin="0,0,0,17" Width="432" Height="78">
                <TextBlock Text="{Binding ProductName}" TextWrapping="NoWrap"
                    Style="{StaticResource PhoneTextExtraLargeStyle}" />
                <TextBlock Text="{Binding Supplier.CompanyName}"
                    TextWrapping="Wrap" Margin="12,-6,12,0"
                    Style="{StaticResource PhoneTextSubtleStyle}" />
            </StackPanel>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

LINQ Support

The client LINQ support in the latest version of the Windows Phone 7 tools—similar to the LINQ support currently available in the .NET client—allows you to write complex LINQ queries and have those LINQ queries translated into a request to the OData service via a URI. The library also supports executing pre-built URIs to an OData service directly, but the LINQ syntax provides a much cleaner experience and hides the complexity of building valid OData URIs. **Figure 5** shows an example of using the LINQ support to create a LINQ expression that will query for all products in the Northwind database that have units in stock.

If you use the Add Service Reference tool described previously, a client proxy context will be generated for you that supports building LINQ queries.

The OData library includes an OData-specific Collection type, `DataServiceCollection<T>`, which implements `INotifyCollectionChanged`. If a `DataServiceCollection` is used to store entities on the client, the entities can be bound to UI elements in XAML and will automatically be change-tracked. Any updates to those tracked entities can be pushed to the service via a call to `SaveChanges` on the client context. **Figure 6** shows some basic XAML binding using the `DataServiceCollection` class—the proxy classes generated using the tools implement `INotifyPropertyChanged` and will reflect the changes made in the UI.

It's possible to turn off the use of the `DataServiceCollection` in the code generation if you're planning to manage the change tracking

manually. This may slightly increase the performance of your app because the notification events won't be raised if this feature isn't used.

The process of executing a query against an OData service and materializing the results involves a call to the network stack on Windows Phone 7, which must always be an asynchronous call. Methods on the `DataServiceContext`, `BeginExecute` and `EndExecute`, asynchronously execute a request to the service using the `IAAsyncResult` pattern. The `IAAsyncResult` pattern can sometimes be cumbersome to use, so a method on the `DataServiceCollection` called `LoadAsync` hides most of the details of the asynchronous execution. The `LoadAsync` method will asynchronously add the result of the query to the `DataServiceCollection` and raise `INotifyCollectionChanged` events for the UI to repopulate itself. The collection also has a `LoadCompleted` event that can be subscribed to, which would be called when the data is done asynchronously loading.

Tombstoning

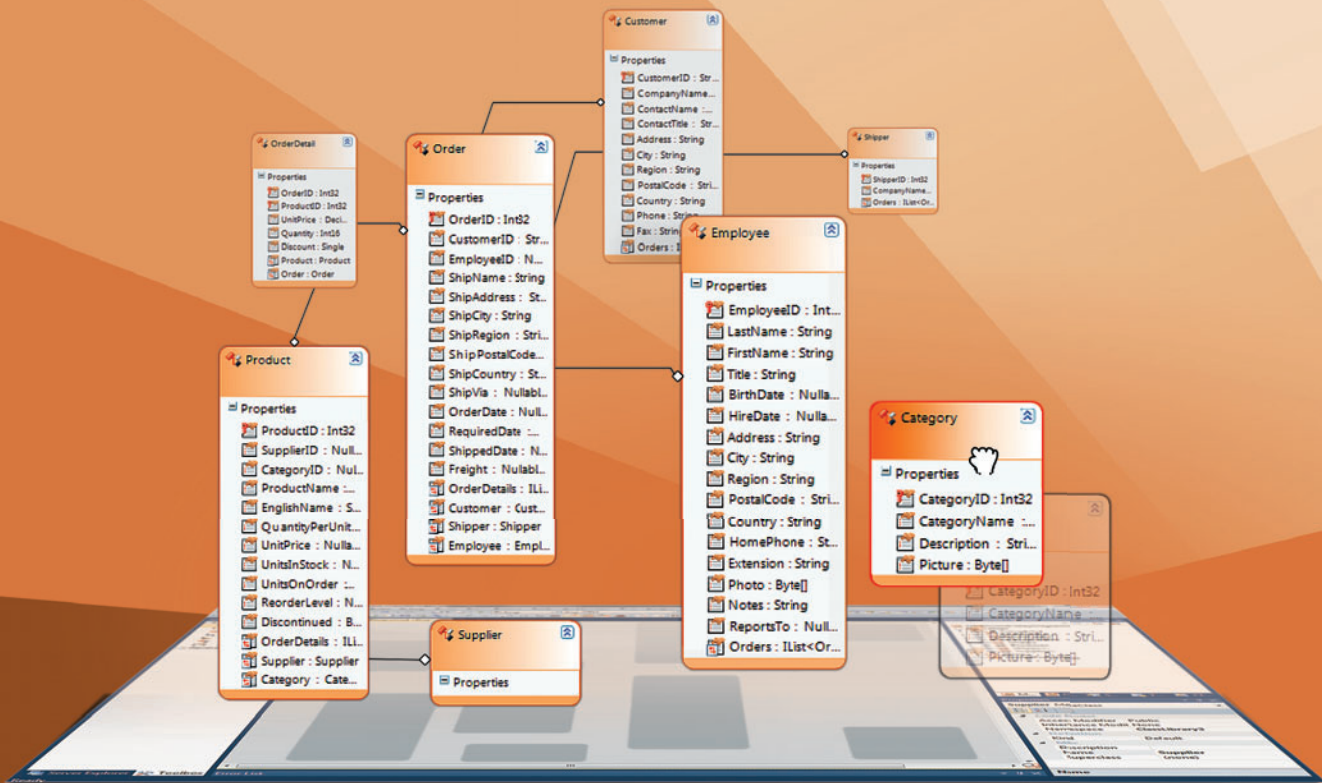
A Windows Phone 7 app will sometimes get a notification from the OS that it needs to deactivate so that the OS can activate another app. This can happen, for example, if the user selects the Home button on the phone or if the user gets an incoming call. The app won't always be deactivated in the Mango beta release of Windows Phone 7 because the fast application-switching feature will often allow the OS to switch to another application without having to deactivate the current one first. When the app is told by the OS to deactivate, the app has the option to store its current state locally so that if it's activated again—which can happen if the user subsequently selects the Back button, for example—the app can restore its previous state and the user can resume what he was doing before the app was deactivated. This is called tombstoning. This can be particularly useful in an app that consumes data from an external data source, because tombstoning saves the app from having to go to the network and use bandwidth to download the data from the external data source again.

Tombstoning in Windows Phone 7 is accomplished by serializing the state of the app into a set of strings in a dictionary. The OData client library for Windows Phone 7 has utilities to aid in serializing the current state by providing methods that can

Figure 7 Serialization/Deserialization of `DataServiceContext`

```
// This method will serialize the local state to be stored
// when the app is deactivated.
public string Serialize()
{
    var saveItems = new Dictionary<string, object>();
    saveItems.Add("Products", ProductsCollection);
    return DataServiceState.Serialize(svcContext, saveItems);
}

// This method will deserialize the local state to be restored
// when the app is reactivated.
public void Deserialize (string state)
{
    DataServiceState dsState= DataServiceState.Deserialize(state);
    svcContext = dsState.Context as NorthwindEntities;
    if (dsState.RootCollections.ContainsKey("Products"))
    {
        ProductsCollection = dsState.RootCollections["Products"] as
            DataServiceCollection<Product>;
    }
}
```



Drag. Drop. Data.

We take care of data access. So you don't have to. Easy as 1, 2, 3.

1. Install **Telerik OpenAccess ORM**.
2. Create a new data model with **drag & drop** (or reuse an *Entity Framework* or *L2S* model).
3. Wire up an UI layer and **run the application**.

Telerik OpenAccess ORM will create the data access code for you, but you can optimize it if you want to.

Try now at: telerik.com/DragDropData

Exclusive offer for MSDN Magazine readers*

Purchase Telerik OpenAccess ORM online and receive a 20% discount.
At checkout, use coupon **DEVPR0M-MSDN2**

* valid until October 31, 2011 for new purchases only.



2011 PARTNER OF THE YEAR
Mobility
Finalist

telerik
deliver more than expected

MSDN Magazine Online

MSDN Magazine

Search MSDN Magazine with Bing 

United States – English ▾ Sign in

Home Topics Issues Columns Downloads Subscribe Contact Us

Digital Magazine Downloads Magazine Blog RSS



August 2011





Team Foundation Server and Exchange: Build a Ticketing System Using Exchange and Team Foundation Server
What do you get if you combine the Team Foundation Server work item tracking functionality with the Exchange Web Services push notification? A unified support ticketing system that integrates e-mail and work items. Mohammad Jalloul shows you how to build it.
Mohammad Jalloul



Visual Studio LightSwitch: Advanced Programming Made Easy With Visual Studio Lightswitch
Visual Studio LightSwitch dramatically simplifies the development of data-centric business applications because it takes care of all the plumbing for you, as Beth Massi illustrates with a sample application.
Beth Massi



Our Name Says It All

activePDF
Leading the iPaper Revolution

Download Today
Call 1-866-GoTo PDF | 949-582-9002 | Sales@activePDF.com

VSIP can help you **build, sell** and **market** Visual Studio extensions



It's like *MSDN Magazine*—only better. In addition to all the great articles from the print edition, you get:

- Code Downloads
- The *MSDN Magazine* Blog
- Digital Magazine Downloads
- Searchable Content

All of this and more at msdn.microsoft.com/magazine



serialize or deserialize a `DataServiceContext` and a set of `DataServiceCollections`. The sample code in **Figure 7** shows an example of using these methods to build a string that represents the current state. Windows Phone 7 applications must implement the `Application_Activated` and `Application_Deactivated` methods that are used by the OS to indicate when the app should activate or deactivate. The serialize and deserialize methods on the `DataServiceContext` can be used from these methods.

Support has been added in the Mango tools to set the `Credentials` property on the HTTP stack when making calls to the OData service.

Credentials

I'll cover one final but important part of building an OData app on Windows Phone 7.1 (Mango): providing credentials to the OData service. In many cases, the service in question is behind an authorization scheme to protect the data from unauthorized use. Support has been added in the Mango tools to set the `Credentials` property on the HTTP stack when making calls to the OData service using the client library. The property takes an implementation of the `ICredentials` interface and passes that interface directly to the HTTP stack. The following code snippet shows the basics of creating the `DataServiceContext` and setting the `Credentials` property using a username, password and domain name:

```
// This method will serialize the local state to be stored
// when the app is deactivated.
public MainViewModel()
{
    // Create the data service context.
    NorthwindEntities serviceContext = new NorthwindEntities(
        new Uri("http://services.odata.org/Northwind/Northwind.svc"));

    // Specify the basic auth credentials.
    serviceContext.Credentials = new NetworkCredential(Username, Password, Domain);
}
```

Learning More

This article gives a summary of OData, the ecosystem that has been built around it and a walk-through using the OData client for Windows Phone 7.1 (Mango) beta to build great mobile app experiences. For more information on OData, visit odata.org. To get more information on WCF Data Services, see bit.ly/pX86x6 or the WCF Data Services blog at blogs.msdn.com/astoriateam. ■

SHAYNE BURGESS is a program manager in the Business Platform Division at Microsoft, working specifically on WCF Data Services and the Open Data Protocol. He regularly blogs on the WCF Data Services blog at blogs.msdn.com/astoriateam.

THANKS to the following technical expert for reviewing this article:
Glenn Gailey

msdnmagazine.com



EXTREME PERFORMANCE

Today's applications need to deliver extreme performance to meet their specs and stay ahead of the curve. As a software architect, you know that fast data access and scalability are the keys to expanding the performance envelope.

ScaleOut StateServer accelerates data access by scaling in-memory storage across multiple servers with blazing speed and industry-leading ease of use. Built-in "map/reduce" parallel analysis provides a powerful computational engine for tracking fast-changing data – with *extreme* performance.

Download your **FREE** trial copy of **ScaleOut StateServer®** today!



SCALEOUT SOFTWARE
Distributed Data Grids for the Enterprise

www.scaleoutsoftware.com/trial | 503-643-3422

No Browser Left Behind: An HTML5 Adoption Strategy

Brandon Satrom

There's a lot to be excited about with HTML5. With new markup, CSS capabilities and JavaScript APIs, the scope of what's possible on the Web is growing by leaps and bounds. Add to that the steady one-upmanship among browser vendors, and the list of exciting features expands almost daily. From nightly builds to dev channel releases and regular platform previews, browsers are changing fast and Web developers everywhere are going along for the exhilarating ride.

But as much as the development and browser communities are pushing the HTML5 hype up to a fever pitch, the vast majority of people on the Web aren't using the brand-new browsers and versions that we are. If you're a Web developer in a large development shop or an enterprise with a large user base, you probably

already know this. Even if you're working with a small shop or startup providing some service via the Web, you probably spend a lot of time making sure your site caters to as many browsers and browser versions as possible.

Given this reality, it's easy to see HTML5 not in terms of whether *it's* ready to be used today, but whether *you're* ready for it. For example, let's suppose you've created a page with some of the new semantic tags (like `<header>` and `<article>`), added some new CSS features (like `border-radius` and `box-shadow`), and even added a `<canvas>` element to draw an HTML5 logo on your page.

In newer browsers like Internet Explorer 9, Firefox 4 and later, or Google Chrome, this will render as depicted in **Figure 1**. But if you attempt to load this page in Internet Explorer 8 or earlier, you'll see something more like **Figure 2**: an utterly broken page.

I wouldn't blame you if you looked at all of the great features of HTML5 and, after having an experience like this, told yourself that the best course was to wait. It's easy to come to the conclusion that, ready or not, HTML5 isn't quite ready for you, or your users.

Before you make the decision to set a date in 2022 to take another look at HTML5, I ask that you read the rest of this article. My goal this month is to give you practical strategies for how you can adopt HTML5 technologies *today* without ending up with the kind of graceless degradation illustrated in **Figure 2**. In this article, I'll cover:

1. Feature detection versus user agent (UA) sniffing
2. Polyfilling with JavaScript
3. Graceful degradation

This article discusses:

- Determining capabilities via feature detection
- Reliable feature detection with Modernizr
- Using JavaScript polyfills to mimic standard APIs
- Using Modernizr to provide graceful degradation

Technologies discussed:

jQuery, JavaScript, Modernizr, PIE

Code download available at:

code.msdn.microsoft.com/mag201109HTML5



Figure 1 A Semantic Page with Styles and an HTML5 <canvas> Element, Rendered in Internet Explorer 9

These three subjects should tell you much of what you need to know to build Web sites for a broad spectrum of browsers. By the time we're finished, you'll have a solid strategy for adopting HTML5 technologies with confidence and without delay. You'll also have some tools you can use to progressively enhance sites for newer browsers, while gracefully degrading for others.

The Importance of Feature Detection

In order to provide stable and consistent experiences across browsers, developers often need to have some information about a user's browser. Historically, it was common to determine that information with JavaScript like this:

```
var userAgent = navigator.userAgent;

if (userAgent.indexOf('MSIE') >= 0) {
    console.log("Hello, IE user");
} else if (userAgent.indexOf('Firefox') >= 0) {
    console.log("Hello, Firefox user");
} else if (userAgent.indexOf('Chrome') >= 0) {
    console.log("Hello, Chrome user");
}
```

This technique, known as UA sniffing, is widely used for determining which browser is requesting your page. The logic is that by knowing the user's browser (Internet Explorer 7, for instance), you can make runtime decisions about what features of your site to enable or disable. UA sniffing is tantamount to saying to the browser: "Who are you?" (For an in-depth analysis on UA sniffing and other detection techniques, see bit.ly/mlgHHY.)

The problem with this approach is that browsers can be made to lie. The UA string is a user-configurable piece of information



Figure 2 The Same Semantic Page, Rendered in Internet Explorer 8 with No Styles and No <canvas>

that doesn't really provide a 100 percent accurate picture of the browser in question. What's more, as this technique became widely embraced, many browser vendors added extra content to their own UA strings as a way to trick scripts into drawing incorrect assumptions about which browser was being used, thus routing around detection. Some browsers now even include a facility that allows users to change their UA string with just a few clicks.

The goal of UA sniffing was never to know the user's browser and version, though. And it certainly wasn't intended to give you an avenue to tell your users to "go download another browser" if they used one you didn't like—even if that technique is used by some. Users do have a choice in the browser they use, but our responsibility as developers is to provide the most reliable and consistent experience, not to impose an opinion of browser preference upon them. The goal of UA sniffing was always to give you an accurate picture of the capabilities or features that you could leverage within the user's current browser. Knowledge of the browser itself is just a means to that information.

Today there are alternatives to UA sniffing, and one that's growing in popularity—thanks in part to both jQuery and Modernizr—is called object or feature detection. These terms are mostly interchangeable, but I'll stick to "feature detection" for this article.

The goal of feature detection is to determine if a given feature or capability is supported on the user's current browser. If UA sniffing is like asking the browser "who are you," feature detection is like asking the browser "what are you capable of"—a much more direct question and a more reliable way for you to provide conditional

Figure 3 A Page with New Semantic HTML5 Markup

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>My Awesome Site</title>
  <style>
    body { font-size: 16px; font-family: arial,helvetica,clear,sans-serif; }
    header h1 { font-size: 36px; margin-bottom: 25px; }
    article
    {
      background: lightblue;
      margin-bottom: 10px;
      padding: 5px;
      border-radius: 10px;
      box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
    }
    article h1 { font-size: 12px; }
  </style>
</head>
<body>
  <header><h1>My Awesome Site</h1></header>
  <article>
    <header><h1>An Article</h1></header>
    <p>Isn't this awesome?</p>
  </article>
  <canvas width="250" height="500"></canvas>
</body>
<script src="../js/html5CanvasLogo.js" type="text/javascript"></script>
</html>
```

functionality to users. It's much tougher for users and browsers to fake or erroneously report feature support, assuming feature detection scripts are correctly implemented.

Manual Feature Detection

So what does feature detection look like, as opposed to the UA sniffing example? To answer that, let's look at fixing the issues that arose when viewing my HTML5 page, depicted in **Figure 1**, in Internet Explorer 8 instead of Internet Explorer 9. The markup for this page is listed in **Figure 3**.

The differences between Internet Explorer 9 and Internet Explorer 8, as shown in **Figures 1** and **2**, are drastic. For starters, my page is completely unstyled, as though the CSS for this page doesn't exist. What's more, I'm missing the fancy HTML5 shield at the bottom of the page. Each of these problems can be fixed easily, and feature detection is the first step to identifying the problem.

The cause of both issues is simple: `<header>`, `<article>` and `<canvas>` are not valid HTML elements as far as Internet Explorer 8 is concerned, and, as such, I can't work with them. To resolve the `<canvas>` issue, instead of using UA sniffing to determine what

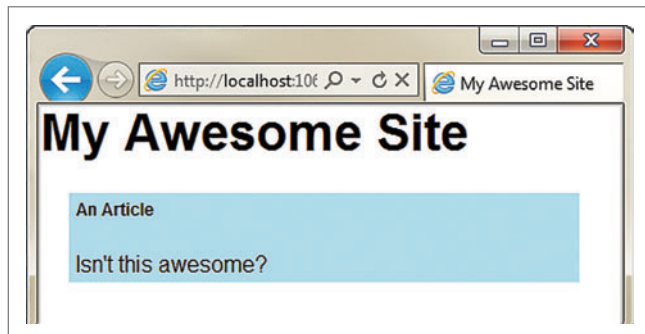


Figure 4 An HTML5 Page in Internet Explorer 8, with the Help of Modernizr

browser/version is being used, I'd like to ask the browser, via JavaScript, if the `<canvas>` element and its JavaScript APIs are supported. My feature check for canvas looks like this:

```
!!document.createElement('canvas').getContext
```

This statement does a couple of things. First, it uses double negation (`!!`) to force undefined values to be explicitly false. Then, it manually creates a new canvas element and attaches it to the DOM. Finally, it invokes `getContext`, a new function available to `<canvas>` elements as a way of manipulating the Canvas API via JavaScript. If I'm using Internet Explorer 9, this statement will return true. If I'm using Internet Explorer 8, `getContext` will return "undefined," which my double negation will coerce to false.

That's feature detection at its most basic. With this statement and others like it, I now have a more reliable way to query feature support within the browser. For more information on manual feature detection, visit bit.ly/9kNDA0.

Using Modernizr for Feature Detection

Manual feature detection is certainly an improvement on UA sniffing, but this approach still leaves you to do the heavy lifting for both detecting the availability of a feature and deciding what to do if that feature doesn't exist. And while the canvas example was a simple one requiring one line of code, this isn't the case for every feature you may want to detect—nor is the detection code the same across all browsers. Detecting support for the CSS3 modules I used earlier (border-radius and box-shadow), for instance, can be a bit trickier.

Thankfully, Modernizr (modernizr.com) offers a better approach. Modernizr is a JavaScript library that "... detects the availability of native implementations for next-generation Web technologies, i.e. features that stem from the HTML5 and CSS3 specifications." Adding a reference to Modernizr in your pages supplies four major features:

1. A comprehensive list of features supported that's cleverly added to your markup, which enables conditional CSS definitions.
2. A JavaScript object that aids in script-based feature detection.
3. All of the new HTML5 tags added to the DOM at run time, for the benefit of Internet Explorer 8 and previous Internet Explorer browsers (more on that in a moment).
4. A script loader for conditionally loading polyfills into your pages.

We won't discuss the first item any further in this article, though I encourage you to head over to Modernizr.com and check out the documentation on this and the rest of the features.

The second item is the feature that helps turn this line of code:

```
!!document.createElement('canvas').getContext
```

Into this line of code:

```
Modernizr.canvas
```

This returns a Boolean indicating whether the canvas element is supported on the page. The great thing about using Modernizr as opposed to rolling your own feature detection is that Modernizr is a well-tested, robust and widely used library that does the heavy lifting for you. Twitter, Google, Microsoft and countless others use Modernizr, and you can too. With the ASP.NET MVC 3 Tools Update (released in April 2011) Microsoft even ships Modernizr in the box with new ASP.NET MVC applications. Of course, all I've accomplished at this point is detecting whether the `<canvas>` element is supported. I haven't said anything about what to do next.

Visual Studio® LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS



MICROSOFT HEADQUARTERS — REDMOND, WASHINGTON | OCTOBER 17-21

BECOME A BETTER DEVELOPER

Join industry experts, Microsoft insiders and fellow members of the .NET community at **Microsoft HQ** for 5 full days of training in a collaborative environment.

- **In-depth training for all levels of developers**
- **60+ sessions – including 18 sessions with Microsoft insiders**
- **9 tracks**
- **5 full-day workshops**
- **Special events and networking opportunities – including VIP access to the Microsoft Campus!**

Sponsored by: **Microsoft®**

REGISTER TODAY

Use promo code MSDNTIP

VSLIVE.COM/REDMOND



Scan the QR code for more information on Visual Studio Live! Redmond.

Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS



OCTOBER 17-21
MICROSOFT HEADQUARTERS
REDMOND, WASHINGTON

5 DAYS OF TRAINING @ MICROSOFT HEADQUARTERS!

- **Visual Studio / .NET**
- **Data Management**
- **Mobile Development**
- **Silverlight/WPF**
- **Developing Services**
- **Cloud Computing**
- **Programming Practices**
- **LightSwitch**
- **Web/HTML5**

SUPPORTED BY:

Microsoft

 **msdn**

 **Microsoft Visual Studio**

Visual Studio
MAGAZINE

Reserve your spot today! Last year's event SOLD OUT.

REGISTER TODAY

VSLIVE.COM/REDMOND

Use promo code MSDNTIP

PRODUCED BY:



PLATINUM SPONSORS



VERSANT



GOLD SPONSORS



MEDIA SPONSOR



Figure 5 Using Modernizr to Polyfill Canvas Support

```
Modernizr.load({
  test: Modernizr.canvas,
  nope: '../js/excanvas.js',
  complete: function () {
    Modernizr.load('../js/html5CanvasLogo.js');
  }
});
```

Figure 6 Using Modernizr and PIE to Add CSS3 Support

```
Modernizr.load({
  test: Modernizr.borderradius || Modernizr.boxshadow,
  nope: '../js/PIE.js',
  callback: function () {
    $('article').each(function () {
      PIE.attach(this);
    });
  }
});
```

Given the knowledge, via feature detection, that a feature is or isn't available to a browser, a common next step is to create some conditional logic that prevents certain code from executing if a feature doesn't exist, or executes an alternate path, similar to this:

```
if (Modernizr.canvas) {
  // Execute canvas code here.
}
```

Adding features to your site based on the presence of additional browser capabilities is referred to as “progressive enhancement,” because you enhance the experience for a more capable browser. At the other end of the spectrum is “graceful degradation,” where the absence of certain features does not cause the browser to error or fail, but rather presents the user with some diminished feature or alternative capability. For older browsers, graceful degradation doesn't have to be your default option. In many cases, it may not even be your best option. Instead, with the assistance of Modernizr, you can often use one of many available browser polyfills to add HTML5-like features to non-supporting browsers.

What Are Polyfills?

According to the Modernizr Web site, a polyfill is “a JavaScript shim that replicates the standard API for older browsers.” “Standard API” refers to a given HTML5 technology or feature, like canvas. “JavaScript shim” implies that you can dynamically load JavaScript code or libraries that mimic those APIs in browsers that don't support them. For instance, a Geolocation polyfill would add the global geolocation object to the navigator object, as well as adding the `getCurrentPosition`

function and the “cords” callback object, all as defined by the World Wide Web Consortium (W3C) Geolocation API. Because the polyfill mimics a standard API, you can develop against that API in a future-proof way for all browsers, with the goal of removing the polyfill once support reaches critical mass. No additional work is needed.

By adding a reference to Modernizr on my page, I do get one immediate polyfilling benefit related to the example in **Figure 3**. The page rendered unstyled because Internet Explorer 8 doesn't recognize tags like `<article>` and `<header>`. And because it didn't recognize them, it didn't add them to the DOM, which is how CSS selects elements to be styled.

When I add a `<script>` tag and reference to Modernizr to my page, the result is a styled page, as in **Figure 4**. I get this benefit because Modernizr manually adds all of the new HTML5 tags to the DOM using JavaScript (`document.createElement('nav')`), which allows the tags to be selected and styled using CSS.

Beyond its use to add support for new HTML5 elements in Internet Explorer, the Modernizr library doesn't provide any additional polyfills out of the box. Those you provide yourself, either from your own scripts or from the ever-growing list of options documented on the Modernizr Web site. As of version 2.0, Modernizr provides a conditional script loader (based on `yepnope.js`—yepnopejs.com) that helps you asynchronously download polyfilling libraries only when needed. Using Modernizr with one or more polyfilling libraries that provide the features you need is a powerful combination.

Using Polyfills to Simulate HTML5 Functionality

In the case of canvas, you can polyfill support for Internet Explorer 8 and earlier with the help of Modernizr and a JavaScript library called `excanvas`, which adds canvas support at the API level to Internet Explorer 6, Internet Explorer 7 and Internet Explorer 8. You can download `excanvas` from bit.ly/bSgyNR and, after you've added it to your script folder, you can add some code to a script block on your page, as shown in **Figure 5**.

Here, I'm using the Modernizr script loader to specify three things:

1. A Boolean expression to test
2. A path to a script to load if the expression evaluates to false
3. A callback to run after the check or script loading is complete

In the context of canvas, this is all I need to add some intelligence and polyfilling to my application. Modernizr will asynchronously load `excanvas.js` *only* for browsers that don't support canvas, and then will load my script library for drawing the HTML5 logo on the page.

Let's look at another example to underscore the value of Modernizr. The detail-oriented among you may have noticed that the site styled in **Figure 4** isn't quite the same as the original page, as rendered in Internet Explorer 9 and depicted in **Figure 1**. The page, as seen in Internet Explorer 8, is missing a box-shadow and rounded corners, and I couldn't possibly ship this awesome site without either, so we'll appeal to Modernizr again for help.

Figure 8 Using Modernizr to Provide Graceful Degradation

```
Modernizr.load({
  test: Modernizr.geolocation,
  yep: '../js/fullGeolocation.js',
  nope: '../js/geolocationFallback.js'
});
```

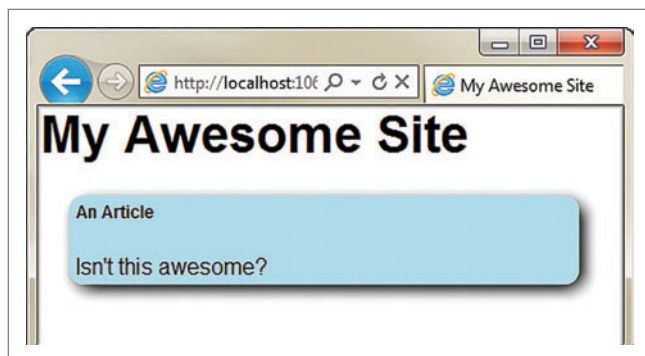


Figure 7 CSS3 Support with Modernizr and PIE

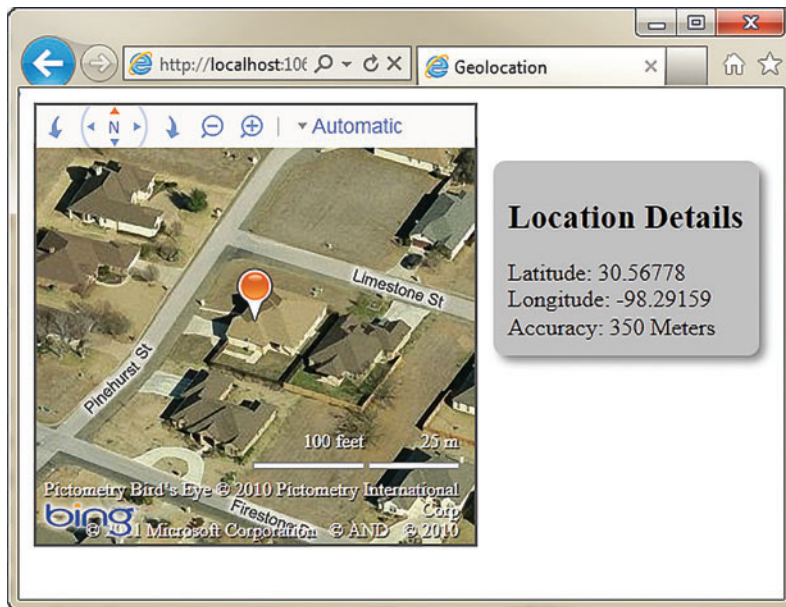


Figure 9 Mapping with Geolocation

As with canvas, Modernizr can tell me that the CSS3 modules aren't supported, but it's up to me to provide a library to polyfill them. Fortunately, there's a library called PIE (css3pie.com) that provides both modules in a single library.

To add support for border-radius and box-shadow, I can add the code in **Figure 6** to my script after I've downloaded PIE. This time, I'll test to see if *either* the border-radius or box-shadow modules

are supported (as opposed to assuming both are always supported or not) and if either is unsupported, dynamically load PIE.js. Once PIE has finished loading, I'll execute a piece of jQuery to select all of my <article> tags and call the PIE.attach function, which adds support for the border-radius and box-shadow styles already defined in my CSS. The end result is depicted in **Figure 7**.

Using Polyfills to Aid in Graceful Degradation

In addition to using the polyfilling techniques discussed here, you can also leverage Modernizr to assist in cases where you wish to gracefully degrade your application, as opposed to polyfilling in another library.

Let's say I have a Bing Maps control on a Web page, and I want to use Geolocation—which we'll cover in depth in a future article—to find the user's current location and then place that location as a pushpin on the map control.

While Geolocation is supported in recent versions of all browsers, it's not supported in older browsers. It's also a little bit trickier to provide full Geolocation API support purely via JavaScript, and even though polyfills for Geolocation do exist, I've decided to instead gracefully degrade my application. In cases where the user's browser doesn't support Geolocation, I'll provide a form she can use to manually enter the location, which I'll then use to position and pin the map.

With Modernizr, it's a simple load call to one of two scripts I've created, as illustrated in **Figure 8**. In this case, I'm testing the Modernizr.geolocation property. If true ("yep"), I'll load my fullGeolocation.js script, which will use the Geolocation API to find my location (with my permission) and place it on a map, as illustrated in **Figure 9**. If, on the other hand, the test is false ("nope"), I'll load a fallback script that displays an address form on my page. When the user submits the form, I'll use the provided address to center and pin the map, as illustrated in **Figure 10**. This way, my page provides a great experience to modern browsers, while degrading gracefully into a reasonable alternative for older ones.

It's easy to look at some of the advanced features of HTML5 and, in the face of a large user base still on older browsers, decide that your site isn't quite ready for them. But there are great solutions already available that not only aid in graceful degradation, but also provide the capability to bring those older browsers into the present so your users can experience HTML5 right now. Over the course of this article, you've seen that with feature detection, Modernizr and polyfilling, it's possible to adopt HTML5 without delay for the ever-growing segment of your users with a modern browser—all the while being careful not to leave the rest of your user base behind. ■

BRANDON SATROM works as a developer evangelist for Microsoft in Austin, Texas. He blogs at UserInexperience.com and can be found on Twitter at twitter.com/BrandonSatrom.

THANKS to the following technical experts for reviewing this article:
Damian Edwards, Scott Hunter and Clark Sell

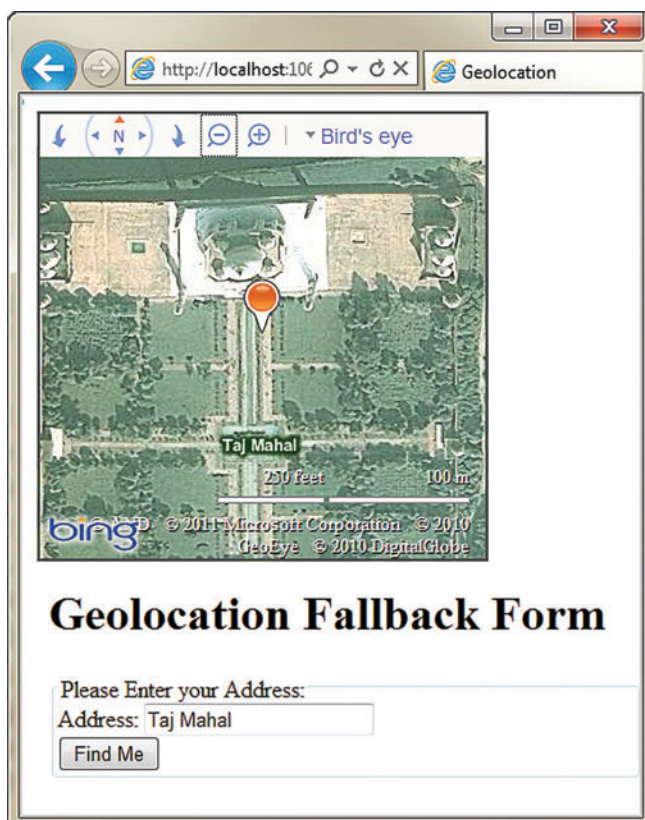


Figure 10 Providing Geolocation Fallback Support



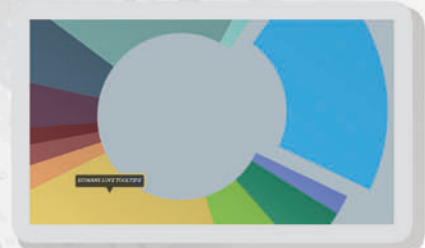
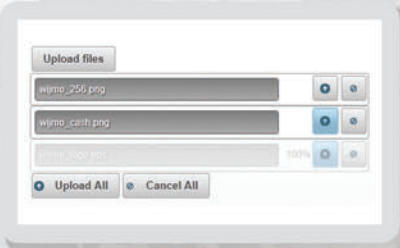
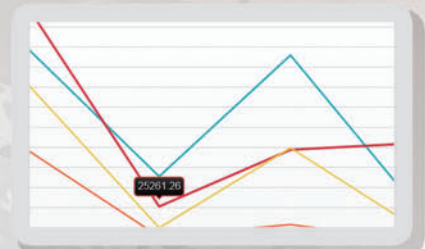
MEET THE NEW WEB STACK

From WebForms to MVC

Our new web stack is the ultimate kit for web development. We have tools that range from WebForms to MVC and from pure client-side to robust server-side development all powered by our core technology: Wijmo.



Number	Player	Age	Birthplace	Position
27	Canada Adams, Craig	32	Saric, Sornal	RW
43	Canada Bouchet, Philippe	36	Saint-Apollinaire, Quebec	D
24	Canada Cooke, Matt	30	Bellville, Ontario	LW
27	Canada Crosby, Sidney (C)	21	Scott's Hill, Nova Scotia	C
1	United States Curry, John	25	Shorewood, Minnesota	G
9	Canada Dupuis, Pascal	30	Laval, Quebec	W
7	United States Eaton, Mark	32	Wilmington, Delaware	D
26	Ukraine Fedchenko, Ruslan	30	Kiev, U.S.S.R.	LW
29	Canada Finlay, Marc-Alexis	24	Sorel, Quebec	G



ComponentOne®
Studio for MVC **wijmo**



Wijmo Scaffolding in MVC
plus Client-side jQuery UI Widgets

ComponentOne®
Studio for ASP.NET **wijmo**



Full-featured ASP.NET Server-side Controls
plus ASP.NET Ajax Extender Controls

DOWNLOAD YOUR FREE TRIALS @
componentone.com/webstack

© 2011 ComponentOne LLC. All rights reserved. All product and company names herein may be trademarks of their respective owners.



Build MVVM Applications in F#

Chris Marinos

Despite being a newcomer to the Visual Studio family, F# has already helped many .NET developers discover the power of functional programming. F# has a strong, growing reputation for its ability to simplify complicated problems like parallel and asynchronous programming, data processing and financial modeling. However, that doesn't mean that F# is a niche language; it's also great for solving everyday problems.

In this article, you'll learn how to use F# to build practical Silverlight and Windows Presentation Foundation (WPF) Model-View-ViewModel (MVVM) applications. You'll see how the same concepts that make F# great for simplifying complicated algorithms can also be used to reduce the ceremony around your view models. You'll also see how the well-publicized asynchronous workflows in F# can be applied in a GUI setting. Finally, I'll cover two common approaches for structuring MVVM applications in F#, as well as the strengths and weaknesses of each approach. By the end of the article, you'll understand that F# is more than just a tool for specialized problems, and you'll be able to use F# to make even the simplest applications easier to read, write and maintain.

This article discusses:

- Reducing ceremony in code
- Using asynchronous programming to solve complicated problems
- Two ways to structure MVVM applications in F#

Technologies discussed:

F#, C#, Silverlight, MVVM

Code download available at:

code.msdn.microsoft.com/mag201109FSharp

Reducing Ceremony

You might think F# is a strange language to use for working with GUI applications. After all, functional languages de-emphasize side effects, and frameworks like Silverlight and WPF are laden with side effects. Maybe you think that view models don't contain enough complicated logic to see a benefit in switching to a functional style. Perhaps you believe that functional programming requires a paradigm shift that makes it difficult to work with design patterns like MVVM. The truth is that using F# to construct view models and models is just as easy as constructing them in C#. View models are also a great place to see how F# reduces ceremonious, boilerplate code. You may be surprised at how effectively F# improves the signal-to-noise ratio in your code if you're used to C# programming.

Figure 1 contains F# code for a simple movie model with fields for a name, a genre and an optional rating. If you aren't familiar with option types in F#, you can think of them as a more powerful and expressive analog to nullables in C#. An option type is a natural way to state that a movie model may or may not have a rating, but it can cause problems for data binding. For example, trying to get the value of an option type set to None (the equivalent of null for nullables) will throw an exception. You may also want to hide the control that's bound to Rating if it's set to None. Unfortunately, if WPF or Silverlight encounters an exception while trying to get the value of Rating, your visibility binding may not execute properly. This is a simple example of where you need a view model to add additional display logic around ratings.

Figure 2 shows a view model with an example of this additional display logic. If the rating is present, its value is passed to the view; otherwise the rating is given a default value of 0. This simple logic prevents exceptions from being raised when Rating is None. The view model also exposes a second property to handle the visibility binding. This method simply returns true if Rating is Some and

false if it is None. The logic in this view model is simple, but the logic isn't the point of the example. Instead, look at how concisely F# expresses the definition of the view model. In C#, view models often become inundated with boilerplate code that clutters the view logic just to make the compiler happy.

Figure 3 shows the same view model written in C#. The increase in ceremonious code is dramatic. It takes roughly four times as many lines of code to express the view model in C# as it does in F#. Much of that increase comes from curly braces, but even significant items like type annotations, return statements and accessibility modifiers get in the way of understanding the logic that the view model is built to encapsulate. F# reduces this noise and places the focus on the code that matters. Functional programming has a bad reputation at times for being overly terse and difficult to read, but in this example, it's clear that F# doesn't sacrifice clarity for brevity.

The preceding example shows one benefit that F# offers when writing view models, but F# also fixes another common problem with view models in MVVM applications. Say that your domain has changed and your model needs to be updated; Genre is now a list of tags instead of a single string. In the model code, that's a one line change from this:

```
Genre: string
```

To this:

```
Genre: string list
```

However, because this property was communicated to the view through the view model, the return type on the view model also needs to change. This requires a manual change to the view model in C#, but it happens automatically in F# due to type inference. This may be surprising if you aren't accustomed to type inference in F#, but it's exactly the behavior you want. Genre doesn't require any display logic, so the view model simply hands this field off to the view without modification. In other words, you don't care about the return type of the property on the view model as long as it matches the return type of the property on the model. That's exactly what the F# code states. Keep in mind that F# is still statically typed, so any misuse of the field on the view model or model (XAML aside) will be a compile error.

Leveraging Existing Assets

The view models in **Figure 1** and **Figure 2** supported only one-way binding because they were only responsible for adding display logic to the model. These simple view models are useful for demonstrating the ability of F# to reduce ceremonious code, but view models usually need to support two-way binding by implementing `INotifyPropertyChanged` for mutable properties. It's common for C# MVVM applications to contain a view model base class to make implementing `INotifyPropertyChanged` and other view model concerns easier. You might be worried that you have to re-implement this behavior in F#, but F# allows you to reuse existing C# view model base classes without having to rewrite them.

Figure 4 shows the use of the `ViewModelBase` class in F#. `ViewModelBase` is a C# base class written by Brian Genisio (houseofbilz.com) that I like to use for all of my C# and F# view models. In **Figure 4**, the base class provides the `base.Get` and `base.Set` functions that are used to implement `INotifyPropertyChanged`. `ViewModelBase` also supports convention-based command generation using the

Figure 1 A Simple Model for a Movie

```
type Movie = {
    Name: string
    Genre: string
    Rating: int option
}
```

Figure 2 A View Model for a Movie with Display Logic for Ratings

```
type MovieViewModel(movie:Movie) =
    member this.Name = movie.Name

    member this.Genre = movie.Genre

    member this.Rating =
        match movie.Rating with
        | Some x -> x
        | None -> 0

    member this.HasRating = movie.Rating.IsSome
```

Figure 3 A C# View Model for Movies

```
class MovieViewModelCSharp
{
    Movie movie;

    public MovieViewModelCSharp(Movie movie)
    {
        this.movie = movie;
    }

    public string Name
    {
        get { return movie.Name; }
    }

    public string Genre
    {
        get { return movie.Genre; }
    }

    public int Rating
    {
        get
        {
            if (OptionModule.IsSome(movie.Rating))
            {
                return movie.Rating.Value;
            }
            else
            {
                return 0;
            }
        }
    }

    public bool HasRating
    {
        get
        {
            return OptionModule.IsSome(movie.Rating);
        }
    }
}
```

Figure 4 Inheriting from a C# ViewModelBase Class

```
type MainWindowViewModel() =
    inherit ViewModelBase()

    member this.Movies
    with get() =
        base.Get<ObservableCollection<MovieViewModel>>("Movies")

    and set(value) =
        base.Set("Movies", value)
```


dynamic programming features of C#. Both of these features work seamlessly with an F# view model because F# was designed to easily interoperate with other .NET languages. Check out the source at viewmodelsupport.codeplex.com for more information on how to use `ViewModelBase`.

Getting Asynchronous

Supporting asynchronous and cancelable operations is another common requirement for view models and models. Fulfilling this requirement using traditional techniques can add significant complexity to your application, but F# contains powerful asynchronous programming features to simplify this task. **Figure 5** shows a synchronous call to a Web service to get movie data. The response from the server is parsed into a list of movie models. The models in this list are then projected into view models and added to an `ObservableCollection`. This collection is data-bound to a control on the view to display the results to the user.

Converting this code to run asynchronously would require a complete overhaul of the control flow using traditional asynchronous

Figure 5 A Sample Web Request for Processing Movies

```
member this.GetMovies() =
    this.Movies <- new ObservableCollection<MovieViewModel>()

    let response = webClient.DownloadString(movieDataUri)

    let movies = parseMovies response

    movies
    |> Seq.map (fun m -> new MovieViewModel(m))
    |> Seq.iter this.Movies.Add
```

Figure 6 An Asynchronous Web Request for Processing Movies

```
member this.GetMovies() =
    this.Movies <- new ObservableCollection<MovieViewModel>()

    let task = async {
        let! response = webClient.AsyncDownloadString(movieDataUri)

        let movies = parseMovies response

        movies
        |> Seq.map (fun m -> new MovieViewModel(m))
        |> Seq.iter this.Movies.Add
    }

    Async.StartImmediate(task)
```

Figure 7 Cancellation in F#

```
let mutable cancellationSource = new CancellationTokSource()

member this.GetMovies() =
    this.Movies <- new ObservableCollection<MovieViewModel>()
    cancellationSource.Cancel()
    cancellationSource <- new CancellationTokSource()

    let task = async {
        let! response = webClient.AsyncDownloadString(movieDataUri)

        let movies = parseMovies response

        movies
        |> Seq.map (fun m -> new MovieViewModel(m))
        |> Seq.iter this.Movies.Add
    }

    Async.StartImmediate(task, cancellationSource.Token)
```

libraries. You'd have to break your code into separate callback methods for each asynchronous call. This adds complexity, makes the code more difficult to reason about and greatly increases the maintenance overhead of the code. The F# model doesn't have these problems. In F#, you can implement asynchronous behavior by making a small set of changes that don't impact the structure of the code, as **Figure 6** shows.

The example in **Figure 6** shows the same code running asynchronously. The code that calls the Web service and updates the result list is wrapped in an `async` block to start the change. The `let!` keyword is used inside this block to tell F# to run a statement asynchronously. In the example, `let!` tells the Web client to asynchronously make a Web request. F# provides the `AsyncDownloadString` method as an extension to `WebClient` to facilitate this process. The last change is the call to `Async.StartImmediate`, which starts the `async` block on the current thread. Running on the GUI thread avoids messy exceptions that happen if you try to update the GUI on a background thread, and the asynchronous behavior ensures that the GUI won't get hung while a Web request occurs.

The change to run this behavior asynchronously didn't require much additional code, but perhaps just as importantly, it didn't require a change in the way the code was structured. When you write the code the first time, you don't have to worry about designing it to possibly run asynchronously in the future. You're free to write synchronous code when prototyping and then easily convert it to `async` when it becomes necessary. This flexibility can save you hours of development time and make your clients much happier when you're able to respond to change more rapidly.

This style of asynchronous programming should look familiar if you've been keeping up with the latest developments in C#. That's because the `async` updates that are coming to C# are heavily based on the F# model. `Async` in C# is available via a community technology preview (bit.ly/qyqgW9), but the asynchronous features of F# are production-ready today. Asynchronous workflows have been available in F# since the language was released, so they're an excellent example of why learning F# will make you a better C# developer.

Though the C# and F# models are similar, there are some differences, and cancellation is a major one. The code in **Figure 7** adds cancellation to the `GetMovies` function. Again, this requires a very small change. To make the workflow support cancellation, you need to create a `CancellationTokSource` and pass its cancellation token to the `Async.StartImmediate` function. **Figure 7** includes some additional setup code at the top of the `GetMovies` function to cancel any outstanding operations to avoid updating the observable collection more than once. A new `CancellationTokSource` is also issued with every call to the function to ensure that each run of the workflow has a unique `CancellationToken`.

In the C# model you have to manually pass a `CancellationToken` down the chain of function calls to support cancellation. This is an intrusive change that could potentially require you to add an extra argument to many function signatures. You also need to manually poll the `CancellationToken` to see if cancellation is requested. The F# model requires much less work. Whenever the asynchronous workflow encounters a `let!`, it will check the `CancellationToken` it received in the `StartImmediate` call. If that token is valid, the

WINDOWS FORMS / WPF / ASP.NET / ACTIVEX

WORD PROCESSING COMPONENTS

MILES BEYOND RICH TEXT



- ➔ TRUE WYSIWYG
- ➔ POWERFUL MAIL MERGE
- ➔ MS OFFICE NOT REQUIRED
- ➔ PDF, DOCX, DOC, RTF & HTML

MEET US AT
SiliconValley
codecamp_11
los altos/ca • oct. 8-9, 2011

TX
TEXTCONTROL®
word processing components

Word Processing Components for
Windows Forms, WPF and ASP.NET

WWW.TEXTCONTROL.COM

Microsoft
Visual Studio
PARTNER

TX Text Control Sales:

US +1 877-462-4772 (toll-free)
EU +49 421-4270671-0

Figure 8 A Dummy F# View Model

```
namespace Core.ViewModels

type MainWindowViewModel() =
    member this.Text = "hello world!"
```

Figure 9 Connecting the F# View Model to the C# View

```
protected override void OnInitialized(EventArgs e)
{
    base.OnInitialized(e);

    this.DataContext = new MainWindowViewModel();
}
```

workflow will execute as usual. If the token is invalid, the operation won't execute and the rest of the workflow won't run. Implicit handling of cancellation is a nice feature, but you aren't constrained to this. If you need to manually poll the `CancellationToken` for a workflow, you can access it with the `Async.CancellationToken` property:

```
let! token = Async.CancellationToken
```

Structuring MVVM Applications in F#

Now that you've seen a few practical ways that F# can enhance your view models and models, I'll explain how to incorporate F# into your Silverlight and WPF applications. There are many different ways to structure your MVVM applications in C#, and the same is true in F#. I'll cover two of these approaches in this article: the all-F# approach, and the polyglot approach that uses C# for views and F# for view models and models. I prefer the polyglot approach for a few reasons. First, it's the approach recommended by the F# team. Second, tool support for WPF and Silverlight in C# is far more robust than it is for F#. Finally, this approach allows you to incorporate F# into existing applications and provides a low-risk way to try out F# in your MVVM applications.

The all-F# approach is nice because it allows you to write your application in one language, but it does come with some limitations. I'll show you how to overcome a few of these common roadblocks a little later, but I find that the rewards aren't worth the workarounds for anything but small applications. The polyglot approach does restrict you from using F# in your view code, but well-constructed MVVM applications should contain very limited amounts of view logic. Furthermore, C# is a great language for writing view logic when necessary because view logic tends to be side-effect-heavy and imperative in nature.

Using the Polyglot Approach

It's very easy to create an MVVM application using the polyglot approach. First, create a new WPF project in C# using the WPF Application project template. This project is responsible for any views and codebehind you need for your application. Next, add a new F# library project to the solution to hold view models, models and any other non-view code. Finally, be sure to add a reference to the F# library project from the C# WPF project. This setup is all that's required to get started with the polyglot approach.

F# is designed for smooth interoperability with any .NET language, so that means you can connect F# view models to C# views using any method you traditionally use for C# view models. I'll show you an

example that uses codebehind for simplicity's sake. First, create a simple view model by renaming `Module1.fs` in the F# project to `MainWindowViewModel.fs`. Fill the view model with the code from **Figure 8**. Connect the F# view model to a C# view using the code from **Figure 9**. If you didn't know that the view model was written in F#, you wouldn't be able to tell the difference from a C# view model.

Add a text box to `MainWindow.xaml` and set the `Binding` to `Text`. Again, everything behaves just as if the application were written exclusively in C#. Make sure the binding works by running the application and viewing the standard "hello world!" greeting.

The All-F# Approach

As I mentioned earlier, I prefer the polyglot approach for its ease of use and flexibility. However, I'll also discuss the all-F# approach to show you the tradeoffs. Visual Studio 2010 ships with a template for creating Silverlight libraries in F#, but it doesn't include any templates for creating WPF or Silverlight applications in F#. Fortunately, there are several great online templates for this. I recommend starting with the templates created by Daniel Mohl (bloggemdano.blogspot.com), because they include sample applications you can use to see the structure of a full application. I'll show you how to build a WPF F# application from scratch for the sake of learning, but I recommend using one of the online templates in practice.

Create a new F# application project called `FSharpOnly` to start with the all-F# approach. Wait for the project to finish being created, then open the project properties and change the output type to `Windows Application`. Now add references to `PresentationCore`, `PresentationFramework`, `PresentationUI`, `System.Xaml` and `WindowsBase`. Add

Figure 10 A Sample `App.xaml` File

```
<Application
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="assembly:FSharpOnly"
  StartupUri="MainWindow.xaml">
</Application>
```

Figure 11 A Sample `MainWindow.xaml` File

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="assembly:FSharpOnly"
  Title="Sample F# WPF Application Written Only in F#"
  Height="100"
  Width="100" >
  <Grid>
    <TextBlock>Hello World!</TextBlock>
  </Grid>
</Window>
```

Figure 12 A Sample `Program.fs`

```
open System
open System.Windows
open System.Windows.Controls
open System.Windows.Markup

[<STAThread>]
[<EntryPoint>]
let main() =
    let application = Application.LoadComponent(new Uri("App.xaml",
        UriKind.Relative)) :?> Application
    application.Run()
```


DEVELOPED FOR INTUITIVE USE

DynamicPDF—Comprehensive PDF Solutions for .NET Developers

ceTe Software's DynamicPDF products provide real-time PDF generation, manipulation, conversion, printing, viewing, and much more. Providing the best of both worlds, the object models are extremely flexible but still supply the rich features you need as a developer. Reliable and efficient, the high-performance software is easy to learn and use. If you do encounter a question with any of our components, simply contact ceTe Software's readily available, industry-leading support team.



DynamicPDF

[WWW.DYNAMICPDF.COM](http://www.DynamicPDF.com)

TRY OUR PDF SOLUTIONS FREE TODAY!

www.DynamicPDF.com/eval or call 800.681.5008 | +1 410.772.8620

 **ceTe software**

files called App.xaml and MainWindow.xaml to the project and set the Build Action on each of these files to Resource. Note that by default there's no item template in F# to generate XAML files, but you can use the general text document template with an extension of .xaml. Fill the XAML files with the code in **Figure 10** and **Figure 11**, respectively. App.xaml and MainWindow.xaml perform the same functions that they do in a standard C# WPF application.

Now add the code in **Figure 12** to Program.fs. This code is responsible for loading the App.xaml file and running the application.

Run the application to get the "Hello World!" greeting. At this point you're free to adopt whatever technique you prefer for wiring your view models to your models, just as you were with the polyglot approach.

One problem you may encounter with the all-F# approach involves static resources. App.xaml is normally defined as an ApplicationDefinition in C# WPF projects, but it's defined as a Resource in the all-F# approach. This causes the resolution of static resources defined in App.xaml to fail at run time when you consume them from other XAML files. The workaround is simple: Change the Build Action of the App.xaml file to ApplicationDefinition and reload the designer. This will cause the designer to recognize

Figure 13 Main.fs Modified to Hook into UI Elements

```
let initialize (mainWindow:Window) =
    let button = mainWindow.FindName("SampleButton") :?> Button
    let text = mainWindow.FindName("SampleText") :?> TextBlock

    button.Click
    |> Event.add (fun _ -> text.Text <- "I've been clicked!")

[<STAThread>]
[<EntryPoint>]
let main() =
    let application = Application.LoadComponent(new Uri("App.xaml",
        UriKind.Relative)) :?> Application

    // Hook into UI elements here
    application.Activated
    |> Event.add (fun _ -> initialize application.MainWindow)

    application.Run()
```

Figure 14 A XamlLoader Class for Creating User Controls in F#

```
type XamlLoader() =
    inherit UserControl()

    static let OnXamlPathChanged(d:DependencyObject) (e:DependencyPropertyC
        hangedEventArgs) =
        let x = e.NewValue :?> string
        let control = d :?> XamlLoader

        let stream = Application.GetResourceStream(new Uri(x, UriKind.
            Relative)).Stream
        let children = XamlReader.Load(stream)
        control.AddChild(children)

    static let XamlPathProperty =
        DependencyProperty.Register("XamlPath", typeof<string>,
            typeof<XamlLoader>, new PropertyMetadata(new PropertyChangedCallback(OnX
                amlPathChanged)))

    member this.XamlPath
    with get() =
        this.GetValue(XamlPathProperty) :?> string

        and set(x:string) =
            this.SetValue(XamlPathProperty, x)

    member this.AddChild child =
        base.AddChild(child)
```

resources in the App.xaml and load your views properly. Don't forget to change App.xaml back to a Resource when you want to build the application, or you'll get a build error.

Codebehind also works differently in the all-F# approach. F# doesn't support partial classes, so XAML files can't be associated with an .fs codebehind file like they can in C#. It's good practice to avoid codebehind whenever possible in MVVM applications, but sometimes codebehind is the most practical way to solve a problem. There are a couple of ways you can work around the lack of traditional codebehind in F#. The most straightforward way is to simply construct your entire view in F#. Although this approach is straightforward, it can also be cumbersome because you lose the declarative nature of XAML. The other approach is to hook into your visual elements when the application is constructed. **Figure 13** shows an example of this approach.

The lack of partial classes in F# also makes it difficult to consume user controls. It's easy to create user controls in XAML, but you can't reference the user controls in other XAML files because there's no partial class definition in the assembly. You can work around this by creating a XamlLoader class, as shown in **Figure 14**.

This class lets you set the path to a XAML file using a dependency property. When you set this property, the loader parses the XAML from the file and adds the controls defined in the file as children of itself. In XAML, this is used as follows:

```
<local:XamlLoader XamlPath="UserControl1.xaml" />
```

The XamlLoader workaround enables you to create user controls without reverting to the polyglot approach, but it's one more obstacle you don't have when creating views in C#.

Closing Thoughts

Now that you've seen F# in action, it's clear that it's a language for writing practical applications. You saw that F# reduces the ceremony in your code, making it easier to read and maintain your view models and models. You learned how to use features like asynchronous programming to solve complicated problems quickly and flexibly. Finally, I showed you an overview of two major ways to structure MVVM applications in F#—and the tradeoffs you experience with each approach. Now you're ready to unleash the power of functional programming in your Silverlight and WPF applications.

The next time you write a Silverlight or WPF application, try writing it in F#. Consider writing portions of an existing application in F# using the polyglot approach. You'll quickly see a drastic reduction in the amount of code you have to write and maintain. Roll up your sleeves and get your hands dirty with the all-F# approach; you'll definitely learn something you didn't know about WPF, Silverlight or F#. No matter what step you take next, once you experience the joy of programming in F#, you won't be able to look at C# the same way. ■

CHRIS MARINOS is a software consultant and Microsoft MVP focusing in F#. He works at SRT Solutions in Ann Arbor, Mich., and is enthusiastic about F# and functional programming. You can hear him speak about these and other interesting topics around the Ann Arbor area or by visiting his blog at chrismarinos.com.

THANKS to the following technical experts for reviewing this article:
Cameron Frederick and Daniel Mohl

Does your Team do more than just track bugs?

Free Trial and Single User FreePack™ available at www.alexcorp.com

Alexsys Team® does! Alexsys Team 2 is a multi-user Team management system that provides a powerful yet easy way to manage all the members of your team and their tasks - including defect tracking. Use Team right out of the box or tailor it to your needs.



Alexsys Team

Track all your project tasks in one database so you can work together to get projects done.

- Quality Control / Compliance Tracking
- Project Management
- End User Accessible Service Desk Portal
- Bugs and Features
- Action Items
- Sales and Marketing
- Help Desk

Native Smart Card Login Support including Government and DOD



New in Team 2.11

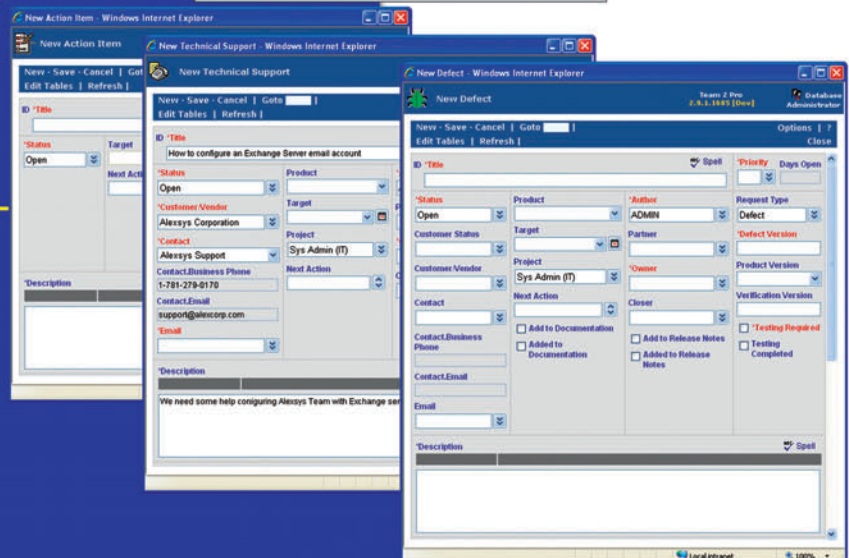
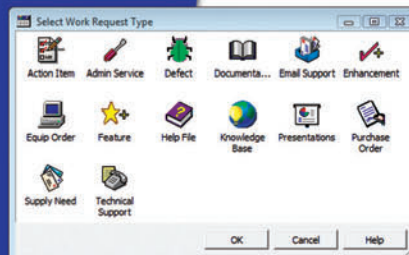
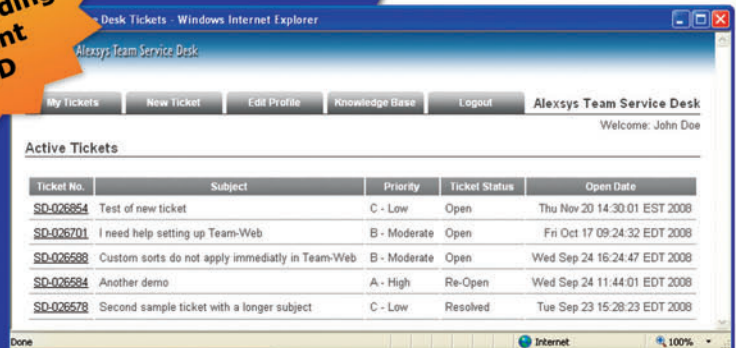
- Full Windows 7 Support
- Windows Single Sign-on
- System Audit Log
- Trend Analysis
- Alternate Display Fields for Data Normalization
- Lookup Table Filters
- XML Export
- Network Optimized for Enterprise Deployment

Service Desk Features

- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder

Team 2 Features

- Windows and Web Clients
- Multiple Work Request Forms
- Customizable Database
- Point and Click Workflows
- Role Based Security
- Clear Text Database
- Project Trees
- Time Recording
- Notifications and Escalations
- Outlook Integration



Free Trial and Single User FreePack™ available at www.alexcorp.com. FreePack™ includes a free single user Team Pro and Team-Web license. Need more help? Give us a call at 1-888-880-ALEX (2539).

Team 2 works with its own standard database, while Team Pro works with Microsoft SQL, MySQL, and Oracle Servers.
Team 2 works with Windows 7/2008/2003/Vista/XP.
Team-Web works with Internet Explorer, Firefox, Netscape, Safari, and Chrome.

Visual Studio ALM Rangers—Reflections on Virtual Teams

Brian Blackman and Willy-Peter Schaub

The **Visual Studio ALM Rangers** have learned valuable lessons about organizing and managing teams with members around the globe—all of whom have various skills, motivations, commitments, project affiliations and restrictions. Here, we'll share our experience and provide guidance for working with teams in less-than-ideal distributed and virtual environments.

To recap, the Rangers are a group of experts who promote collaboration between the Visual Studio product group, Microsoft Services and the Microsoft Most Valuable Professional (MVP) community by addressing missing functionality, removing adoption blockers, and publishing best practices and guidance based on real-world experience.

In this article, we'll start by defining our customized project-management process called Ruck; then we'll offer a summary of virtual team challenges and share observations made with Rangers projects using Ruck with Visual Studio Team Foundation Server (TFS) as the Application Lifecycle Management (ALM) solution. We'll conclude with our recommendations for the Rangers and other virtual teams in similar environments.

This article discusses:

- Challenges of dispersed teams
- Lessons learned from recent projects
- The impact of culture and different working conditions
- Dealing with different motivations and commitments
- The top seven recommendations for virtual teams

Technologies discussed:

Visual Studio Team Foundation Server, MSF for Agile Software Development v5.0

Ruck?

The Rangers have been dog-fooding Visual Studio TFS as an ALM solution, with the goal of improving the quality of our business operations and solutions production. One core area of investment with huge potential impact has been the process model and the associated requirements-management process. Our core intent is to create a repeatable and systematic way of finding and addressing the killer features of the Ranger initiatives. Defining and evolving the process has proven to be a challenging task—somewhat like fixing an airplane engine during flight—but with several project teams successfully using variations of the process, we can regard it as a solid pillar. To ensure that we don't confuse anyone in terms of the methodologies or annoy process zealots, we provisionally chose to call our customized process “Ruck,” which, in the rugby world, means “loose scrum.”

Challenges of the Virtual Ranger Team Environment

A frequent challenge with Rangers is that each individual has to balance his work, home, and Ranger worlds and commitments. Most Ranger activities have the lowest priority of these three and often happen late at night. What makes the Rangers ecosystem unique is that our solutions can't be released when they're ready, but are implicitly time-boxed with technology milestones and demand for the solution. As we've yet to find a magic way of extending the number of hours in a day, we rely on an individual Ranger's sheer passion for the technology and the community to help find spare cycles for Ranger activities. Although this creates a heroic IT professional who can absorb numerous context switches and produce high-quality deliverables in short, ad hoc and random spurts, it also introduces a form of anti-Scrum process.



Figure 1 Typical Rangers Meeting Across Time Zones

Looking at the various locations of our team members in the Rangers Index (bit.ly/9LKgZb), which currently includes only a small percentage of the Rangers, we realized that Rangers, and thus our teams, are scattered around the planet. We have to be cognizant that we're working with a variety of different cultures and that we can neither make assumptions about cultural norms or customs, nor take the environments for granted. Thus, we implement situational leadership and process in our projects to accommodate cultural differences (see the book, "Leadership and the One Minute Manager: Increasing Effectiveness Through Situational Leadership" [Morrow, 1985], by Kenneth H. Blanchard, Patricia Zigarmi and Drea Zigarmi).

As shown in **Figure 1**, the vast range of time zones necessitates that some team members occasionally stay up late or get up at odd hours for a status or design meeting. This is neither fun nor productive.

The combination of Rangers from different ecosystems such as product groups, services, partners, MVPs and communities introduces a variety of motivations and commitments that all need to be embraced by the Rangers ecosystem and recognition program.

While we aim for maximum transparency and access to everything by everyone, we have edge cases because of non-disclosure agreements, licensing and infrastructure restrictions.

The frequent context switching—and starting and stopping of a project or feature area—is the most challenging problem because it's a demoralizing experience for the team. This can make sprint burn-down charts interesting to look at, but difficult to use effectively for status and progress tracking, as you can see in **Figure 2**. The first chart in **Figure 2** depicts an increase in task work items as an individual or team learns there's more to implementing the product backlog item than first realized. The second chart depicts stopping work on a project because of customer commitments. The velocity chart in **Figure 3** demonstrates our state-

ment regarding the effects of context switching and the starting and stopping of feature areas.

Finally, because of the voluntary nature of the Rangers program, accountability varies greatly, offering few options to enforce a delivery commitment. Fortunately, most Rangers hold themselves accountable and therefore most Ranger teams deliver on time, every time.

Analyzing and Learning from Recent Projects

As mentioned, we have several challenges when teams aren't located at a single site, when teams are in different geographies and when there's no individual commitment beyond one's passion to be a contributing Ranger. We've attempted to adapt to this by organizing teams by geography, but this often doesn't work because it goes against our belief in letting developers choose work from areas in which they have interest, even when it doesn't align with their current geography. Therefore, we have to be self-organizing and rely on experience to determine what best practices will work for a particular project.

The nature of our virtual ecosystem forces us to break numerous predefined rules of processes such as Scrum. As in Kanban (see the book, "Kanban: Successful Evolutionary Change for Your Technology Business" [Blue Hole Press, 2010], by David J. Anderson), we tailor our process to meet our needs. In many ways, we apply situational

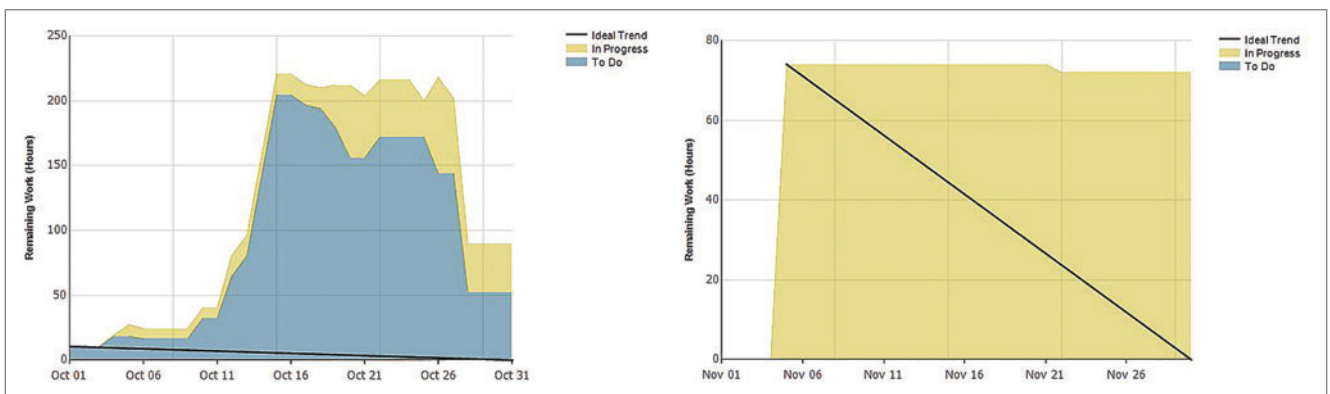


Figure 2 Sprint Burn-Down Charts Reflect Challenges

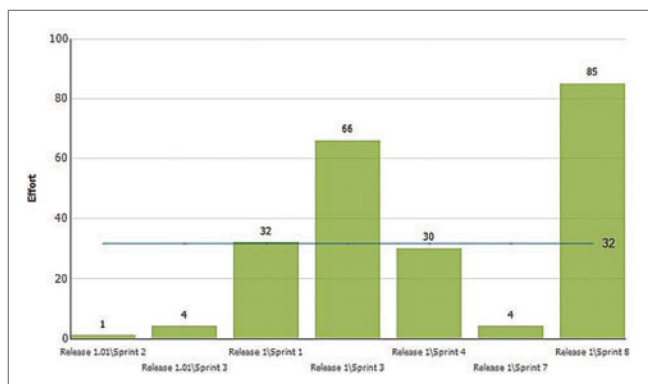


Figure 3 Project Velocity

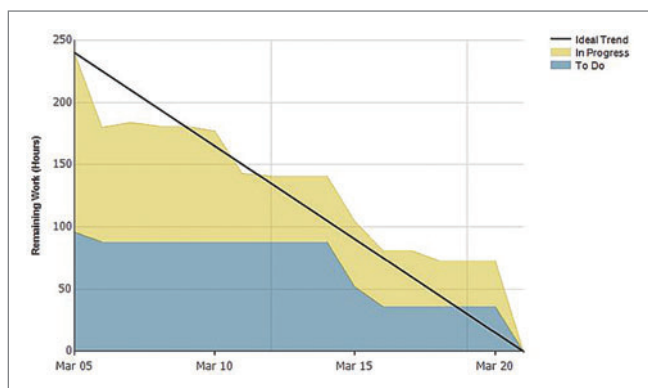


Figure 4 Better Cadence Results in Better Burn-Down

leadership to projects, depending on the unique attributes of each specific project. For example, we conduct weekly meetings because everyone has day jobs. Daily meetings (“stand-ups”) are too bounded and too frequent. In addition, our meetings aren’t classic stand-up Scrum meetings because too few individuals can meet at the same time. This regularly compels us to have two meetings in a single day to accommodate multiple time zones.

Ranger projects are complex, requiring that we have transparency, frequent communication, checkpoints and meetings. Because everyone has other commitments, such as families and their day jobs, teams need to avoid being too bounded. As we learn and adapt, we establish our own project patterns and adapt them for better influence. Some of these patterns include:

- Having team members sign up for the task in which they have interest
- Timely e-mail reminders of key checkpoints or milestones with work progress visualization
- Weekly meetings

Our aforementioned challenges exacerbate as project rhythms change. This isn’t unique to our Ranger ecosystem; we experience this in many projects at startup. Getting a proper cadence with a new team takes a few steps and missteps before everyone is finally in sync, executing with regular rhythm.

We, the authors, haven’t been involved directly in the large open source community outside of Microsoft, so what we write here is based on assumptions that are supported by other publications and blog posts. Unlike open source projects, Ranger projects have a fixed ship-date schedule mindset. As we understand it, most open source projects ship when they’re done or are good enough to release, however long this takes. For Rangers projects, many people depend heavily on our artifacts. If our development cycle takes too long, our work can become obsolete with a new release of Visual Studio. Having a fixed ship-date driving the project could be considered anti-Scrum. However, we don’t see this as different from time-boxing sprints; we time-box the overall project.

For our Ranger projects, we use either the MSF for Agile Software Development v5.0 or the Microsoft Visual Studio Scrum 1.0 process template; the latter is by far more popular because it’s lightweight. Our projects require some up-front planning and design; some process methods refer to this as the pregame. We use a Sprint 0 for this pregame work to set a stake in the ground and time-box the planning and design. This ensures the development start date and objectives are known. During Sprint 0, we conduct required team training, set up the environment, plan, research, vote on epic priority and review our Ruck process (we’ll discuss epics in more detail later). Setup of environments is extremely important when you consider a product such as Microsoft Lab Management.

As in Scrum, our Ruck process requires frequent checkpoints. As discussed earlier, daily meetings are too much for the Ruck team. Our version of a stand-up is a weekly meeting, or series of meetings in different time zones, conducted in Microsoft Lync. These meetings are conducted in classic fashion, where attendees are asked about what they worked on, what they’ll work on next and if there are any impediments. In addition, we use the opportunity to communicate any key messages and reiterate the project/sprint vision. We’ve learned that the use of visuals is most effective. The Ruck master or the development lead will share his desktop to display the sprint burn-down charts and show the list of work items for the current team member to address. Visuals and visibility are key!

Ranger projects often have difficulty achieving consistent velocity, as you can see in **Figure 3**. This is because of the anti-Scrum nature of our teams where team capacity is always in flux. Team members can depart a project at a moment’s notice, leaving us to manage creatively to get work done or solicit new contributors. Because this is

Query Results: 6 items found (1 currently selected).

ID	Work Item Type	Title	Assigned To	State	Area Pa
8294	Product Backlog I...	Epic - Visualisation of the guidance using quick reference posters	Willy-Peter Schaub	Committed	AHE\Co
8488	Product Backlog I...	Epic - Advanced golden image management using the VM Factory for Lab Management	Paul Meyer	Committed	AHE\VN
8489	Product Backlog I...	Epic - Provide guidance on setting up Test environments with respect to pre-defined personas	Harish Reddy Kothapalli...	Done	AHE\Te
8490	Product Backlog I...	Epic - Provide Guidance to enable large and small teams to setup and configure both automated and manual te...	Tony Feissle	Committed	AHE\Mz
8491	Product Backlog I...	Epic - Provide practical guidance for managing and maintaining a Lab Management environment	Mark Nichols	Done	AHE\LaI
8493	Product Backlog I...	Epic - Provide practical guidance to enable teams to quickly setup and configure their lab management environ...	Chris Burrows (AUSTRA...	Committed	AHE\Pla

Figure 5 Use of Epic Prefix in the Title of User Story Work Item Type

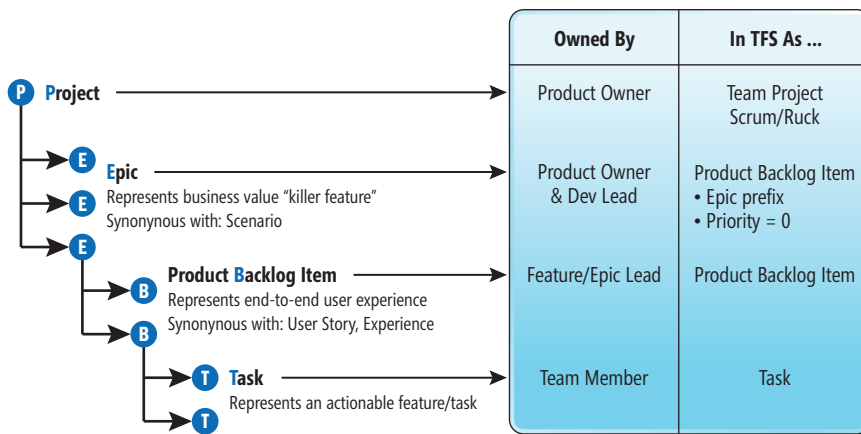


Figure 6 Mapping the Ranger Epic/PBI/Task Hierarchy to Team Foundation Server Artifacts

a common circumstance for our Ranger projects, we've instituted a Ranger Manifesto to emphasize the importance of making the required commitments when contributing to a Ranger project.

We've learned that smaller projects and smaller team sizes enable better execution, proper cadence and deeper team member commitment. Also, it seems it's more difficult for someone to abandon a small team than a larger team. **Figure 4** demonstrates a sprint burn-down chart where we've achieved good cadence with work item completion and where the actual trend of burn-down better matches the ideal trend as compared to the previous burn-down charts shown in **Figure 2**.

We use epics to outline a high-level view or elevator pitch for a feature. This maps closely to the Agile community's use of epics: to tell a more-encompassing story for a feature because the customer isn't on-site to provide details without any time lag. You can look at an epic as a story that tells why someone would want to install our solution or download our guidance. The epic guides the vision for a feature and drives the creation of related user stories from which we create our work. Because the out-of-the-box TFS process templates

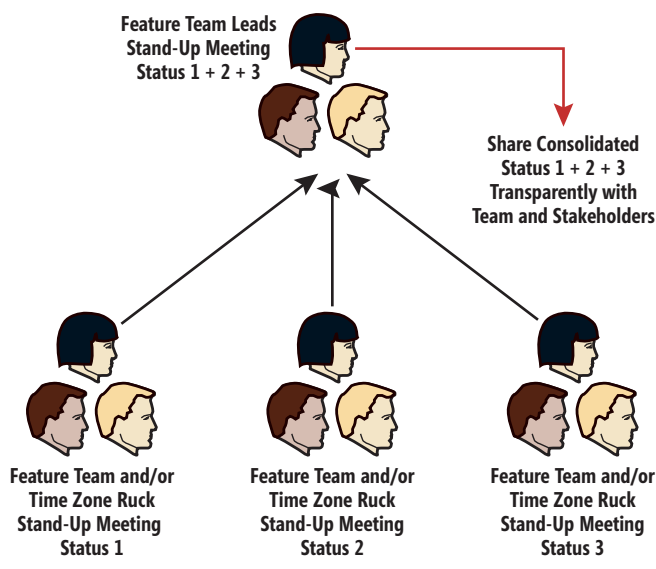


Figure 7 Ruck-of-Ruck Stand-Up Meetings

don't have an epic work item type, and we have constraints where we avoid template customization, we prefix the User Story Title field with "Epic" (see **Figure 5**) and we give this work item a priority of zero for easy identification in reports or report filters.

Ranger projects are represented by multiple, loosely defined roles, such as developer, tester, Ruck master, developer lead, product owner, epic lead and reviewer. In many cases, a team member may assume more than one role. One person could be a developer for one feature, a tester for a different feature and a reviewer for a different project. Three core Rangers usually play steadfast roles on every project: Bijan Javidi as project manager, Willy-Peter Schaub as developer

lead, and Brian Blackman as Ruck master and coach.

The need for frequent checkpoints and complete transparency makes communication a pivotal part of our process and ecosystem. Good communication creates supportive, monitorial and collaborative webs among the geographically dispersed team. To create a sense of belonging, we always have virtual face-to-face kick-off meetings, during which we define a common set of goals, objectives, motivations, visions, benefits, scopes and constraints. Most importantly, during this kick-off meeting, the team agrees on infrastructure and communication guidelines.

Ruck-of-Ruck meetings, discussed later, minimize the need for everyone to be present in weekly stand-up meetings and ensure that the status and impediments are transparent, using ad hoc, spontaneous and predefined communication. Using TFS Work Item Tracking (WIT), we capture information and state such information as the progress, backlog, features, acceptance criteria, bugs and impediments. Stand-up minutes are documented in wikis and design information in documents stored on the common portal. In some cases, the minutes are stored in source control, which—like all other infrastructure components—is accessible from anywhere in the world.

The primary vehicles for communication and collaboration are the shared portal and, most importantly, e-mail. Using distribution lists and a controlled vocabulary, e-mails are shared with everyone, including Rangers outside of the current project scope. This ensures that all stakeholders receive all the information, which drives transparency and, using e-mail rules, delegates selective reading to those interested in the information.

Understanding and Dealing with the Impact of Cultural and Working Conditions

Communication and management styles differ in many countries. Some cultures prefer formal and distant communication, with clearly defined management roles and expectations. Others engage in informal, in-the-face and often collaborative management, where even an embracing "hug" is a sign of respect. To clearly define the potential issues and ensure that there are no misunderstandings that could lead to bitter and often disconnected team members, we have to recognize this diversity. As with many other ALM and

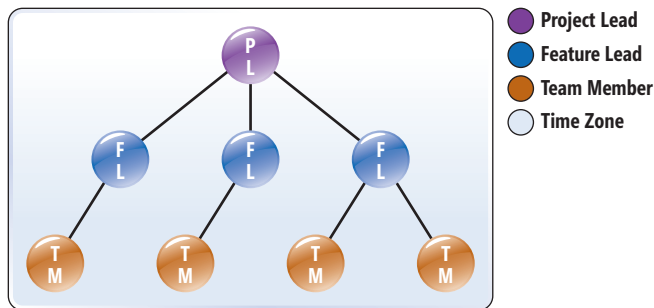


Figure 8 Ideal Virtual Team

virtual team tenants, transparency is crucial to proactively and continuously dealing with the culture challenges.

Time zone differences and cultural differences often go hand-in-hand. While some cultures or individuals don't mind working 24x7—or, more importantly, getting up at insane hours of the night to attend a team meeting—others will take great offense when asked to invest personal family time in a project. To create an environment in which team members feel comfortable and able to collaborate, it's important to find mechanisms and times that are conducive for everyone in the team to participate in conference calls or online discussions. The majority of the Ranger communication and collaboration occurs through e-mail and, more importantly, the TFS team project and associated portal. We also take great care to schedule regular team stand-up meetings where we can promote collaboration, share status and continuously foster the team identity. When we schedule multiple stand-up meetings across time zones, team members aren't restricted to their own time zone. They're allowed to join one or more meetings that suit their time zone and their business commitments. Ruck-of-Ruck stand-up meetings aggregate the status upward, as shown in **Figure 7**, using the collaboration and especially the WIT infrastructure. Once completed, one consolidated team status can be transparently shared with the entire team and shareholders, using the communication mechanisms chosen by the team.

Figure 8 shows the ideal virtual team, operating in one time zone, split into feature areas in which one passionate feature lead works with a dedicated and focused number of team members.

In reality, the typical Ranger team resembles **Figure 9**, where the team is scattered across a number of time zones around the globe. The team leads work with part-time resources in various time zones, often focused on several feature areas within the project.

By giving everyone access to everything, driving complete transparency, fostering a no-nonsense, no-politics team environment and making use of state-of-the-art ALM technology, we've been able to create effective and passionate project teams. As shown in **Figure 10**, all Rangers have

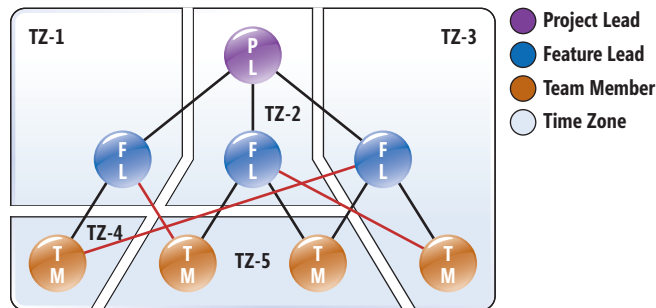


Figure 9 Typical Rangers Virtual Team

access to all projects through one Rangers portal, which includes work items, source control, project portals and the overall Rangers portal. The Rangers use the controlled vocab to categorize e-mails, allowing easy filtering and use of rules.

The diversity of working conditions is probably the most challenging part of our virtual teams. Not only are Rangers part-time team members and therefore regularly disconnected during regular business hours, they often work in secure environments, limiting their access to the public domain and the Rangers ALM infrastructure. In addition, some Rangers work in less-than-favorable environments where 9,600 bps communication lines or 56KB modems are still standard infrastructure. This limits their ability to use environments and services that rely on high-speed networking and huge bandwidth.

Working with TFS—in particular the Web Access components—and relying on e-mail for collaboration, we've been able to create an effective and reliable ALM infrastructure that suits the majority of the Rangers.

Understanding and Dealing with Motivations and Commitments

What are the driving forces that make IT professionals sacrifice personal time to collaborate with other Rangers and aggregate

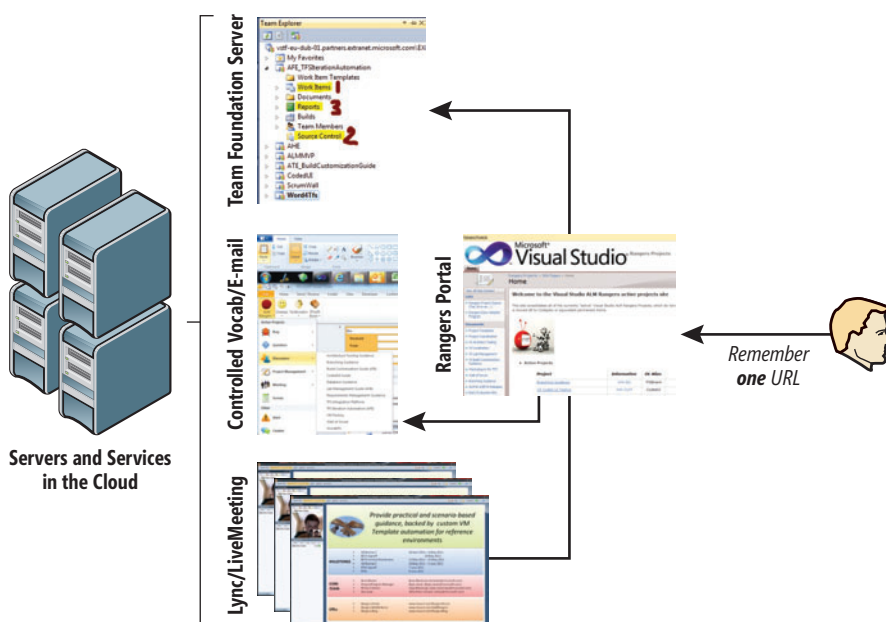


Figure 10 Typical Rangers Infrastructure



SAVE THE DATE!

VISUAL STUDIO LIVE! COMES BACK TO ORLANDO

INTENSE TRAINING + AN AWESOME LOCATION:

that is the Visual Studio Live! experience. You'll get one-on-one access to industry pros over 5 days of real-world education on:

- ▶ **Visual Studio 2010 / .NET**
- ▶ **Silverlight / WPF**
- ▶ **Web / HTML5**
- ▶ **Windows Phone 7**
- ▶ **Data Management**
- ▶ **Windows Communication Foundation**
- ▶ **Cloud Computing**
- ▶ **Programming Practices**

**REGISTRATION
IS NOW OPEN.**

**REGISTER
TODAY TO
SAVE \$300!**

ALL-INCLUSIVE CONFERENCE + WORKSHOP + HOTEL PACKAGES
WITH ACCOMMODATIONS AT UNIVERSAL STUDIOS ROYAL PACIFIC RESORT ARE AVAILABLE!

VSLIVE.COM/ORLANDO

PLATINUM SPONSOR

SUPPORTED BY

PRODUCED BY



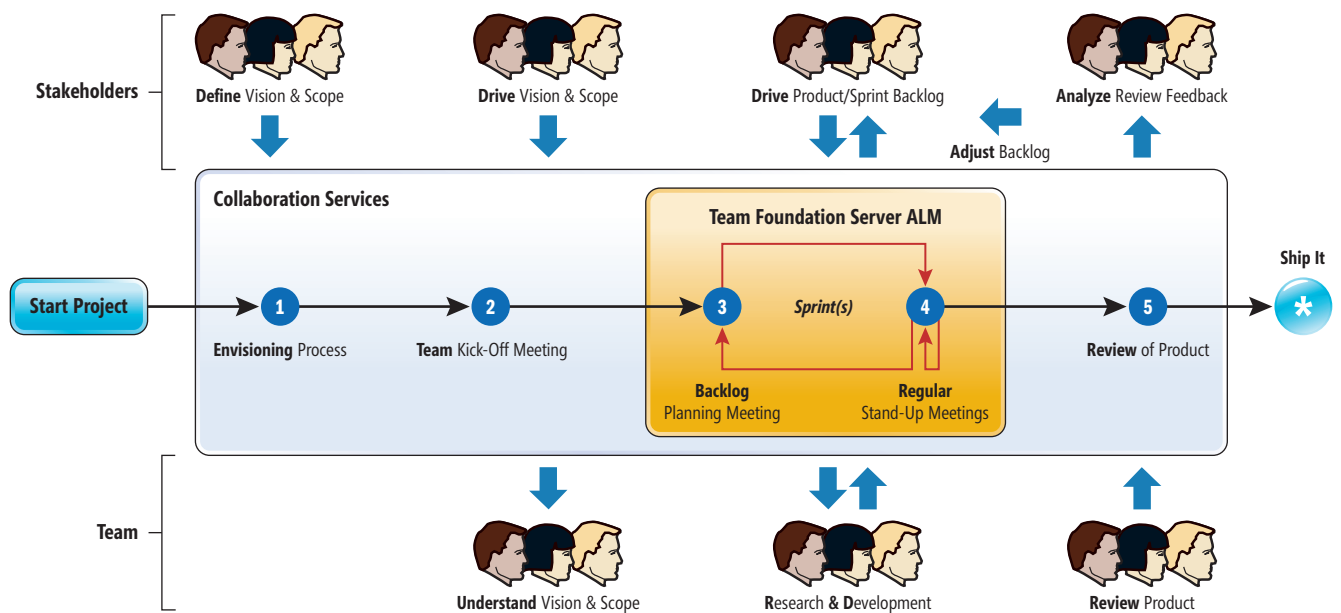


Figure 11 Using Technology for Collaboration and ALM

experience, knowledge and information into Rangers solutions and guidance, often working in isolation? Every individual has a different set of motivations and resultant commitments, which needs to be identified and understood so that each resource can be utilized effectively within the team.

When working with virtual teams, we found that it's important to identify all the team members who are passionate for technology, have a "get it done" mindset and are motivated, open-minded and highly self-disciplined. No team identity, technology or process will enable people who need a great deal of supervision to work effectively in virtual teams, especially when they're also expected to fulfill a team lead role.

Using the Ruck process and the TFS ALM solution to define epics, product backlogs and tasks has proven invaluable, especially when team members are asked to volunteer for tasks, rather than being assigned tasks. In addition, the ability for us to continuously track progress—or the lack thereof—has allowed us to identify and deal with challenges and impediments early. Unfortunately, a challenge we haven't yet conquered is getting around the human factor of getting everyone to regularly update their work items in a timely fashion without constant reminders.

As shown in **Figure 11**, we use a combination of regular team meetings, an ongoing collaboration strategy and technology to define and communicate the team purpose, vision, product backlog and project status with the team and stakeholders. This ensures we get continuous buy-in and commitment from everyone, even when working in a disconnected, virtual, isolated and lonely ecosystem.

It's amazing how an isolated problem becomes a team challenge when its existence and potential impact on the team objective is transparent. Using TFS and our Ruck process has allowed us to keep the status and challenges "in the face of everyone." The result is that the team often identifies and resolves challenges before having to report them at the next weekly stand-up meeting.

Our Top Recommendations

There are many recommendations, skills and practices that will help you improve the virtual team environment, the collaboration and effectiveness of each team member, and the quality of the deliverables.

For us, the seven top recommendations include:

1. Clearly define and foster a team identity and vision.
2. Clearly define the process, such as the Ruck process.
3. Clearly define the communication guidelines and protocols.
4. Clearly define milestones, share status and celebrate the deliverables.
5. Implement an ALM infrastructure that allows the team to collaborate effectively, such as TFS and associated technology.
6. Promote visibility and complete transparency on a regular cadence.
7. Choose your teams wisely, asking members to volunteer and promoting passionate and get-it-done individuals.

In a future article, we'll investigate how the Rangers are using VM Factory and the Visual Studio testing tools in the distributed and virtual team environment to promote consistency and raise the overall quality of our out-of-band solutions. ■

BRIAN BLACKMAN is a principal consultant with the Microsoft Services Partner ISV team, focusing on affecting ISV partners' success in engineering and in the marketplace. He has an MBA and is a CSM, MCSD (C++), MCTS and Visual Studio ALM Core Ranger. He spends his time writing code, creating and delivering workshops, and consulting in various concentrations and all things ALM.

WILLY-PETER SCHAUB is a senior program manager with the Visual Studio ALM Rangers at the Microsoft Canada Development Center. Since the mid-'80s, he's been striving for simplicity and maintainability in software engineering. His blog is at blogs.msdn.com/b/willy-peter_schaub and you can follow him on Twitter at twitter.com/wpschaub.

THANKS to the following technical experts for reviewing this article:
Ben Amodio, Mike Fourie, Bill Heys, Bijan Javidi and Patricia Wagner

TECHMENTOR
CONFERENCES

LAS VEGAS

October 10-14, 2011

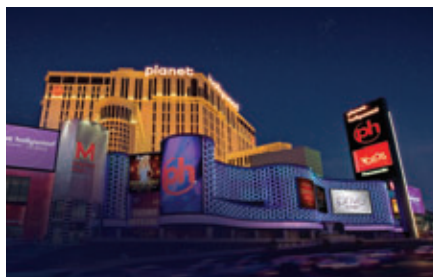
**Planet Hollywood
Resort & Casino**

IN-DEPTH TRAINING FOR IT PROS

Unleash the IT Superhero in YOU at TechMentor

TechMentor Las Vegas is a 5 day event for IT Professionals who are seeking real-world, in-depth and unbiased technical training. This boutique conference will provide you with immediately-useable information that can help you conquer any IT obstacle and master Windows Server 2008 R2 and Exchange 2010.

- » Connect with IT Experts
- » 50+ Sessions
- » 8 Focused Tracks
- » Deep Dive Content
- » Extended MCITP Boot Camp
- » Tutorials



REGISTER TODAY: TECHMENTOREVENTS.COM/LV

GROUP RATES AND DISCOUNTS AVAILABLE. USER CODE: TMAD2

SUPPORTED BY



PRODUCED BY



MEDIA SPONSOR





Multiparadigmatic .NET, Part 10: Choosing an Approach

In my last column (Part 9 of this series), I suggested that any time an article series gets close to double digits, the author is either pretentious enough to think his readers are actually interested in that subject that many times in a row, or he's just too boneheaded to come up with a new topic. Readers are left to speculate as to which of those two cases are at work here.

Nevertheless, commentary from the field has made it clear that despite the dangers of traversing into double-digit territory, one more article on multiparadigmatic design seems necessary to try to tie all the individual elements together—to demonstrate how to use each of these different paradigms and choose among them for a real-world problem. “Real-world,” for our purposes, means non-trivial enough to hint at how the approach might be useful for problems that aren't as simple as those chosen to solve in a magazine article.

Trying to create such a problem turns out to be more difficult than it might appear; either the idea is too complicated, with too many distractions in it to get a clear view of the solutions used; or the idea is too simple, allowing for too little variation in its implementation to illustrate how the different paradigms might each be used to solve it. Fortunately, to get started we can make use of some work that has already been done—in the form of Dave Thomas' “Code Katas.”

Code Katas

On his Web site (codekata.pragprog.com), Thomas writes about dropping his son Zachary off at karate practice and discovering that the dojo had no room left in the parents' viewing area for him. This left him with 45 minutes to himself, and he began to play around with some code he'd been idly considering performance implications about. He writes:

I just wanted to play with some code and experiment with a technique I hadn't used before. I did it in a simple, controlled environment and I tried many different variations (more than I've listed here). And I've still got some more playing to do ...

What made this a practice session? Well, I had some time without interruptions. I had a simple thing I wanted to try, and I tried it many times. I looked for feedback each time so I could work to improve. There was no pressure: the code was effectively throwaway. It was fun: I kept making small steps forward, which motivated me to continue. Finally, I came out of it knowing more than when I went in.

Ultimately, it was having the free time that allowed me to practice. If the pressure was on, if there was a deadline to deliver the blog search functionality, then the existing performance would have been acceptable and the practice would never have taken place. But those 45 pressure-free minutes let me play.

So, my challenge for the day: See if you can carve out 45 to 60 minutes to play with a small piece of code. You don't necessarily have to look at performance; perhaps you could play with the structure, or the memory use or the interface. In the end, it doesn't matter. Experiment, measure, improve.

In other words, the code kata is a (relatively) simple problem—one that's not hard to grasp conceptually—that offers a framework in which to explore. In our particular case, the goal will be to design. We'll base our exploration on Thomas' “Kata Four: Data Munging.”

Kata Four: Data Munging

The code kata in this case has three parts, which we will take in discrete steps, one at a time, designing as we go and considering each of the axes available to us within C# (though, again, solutions in Visual Basic are equally approachable).

A code kata is a (relatively)
simple problem that offers a
framework in which to explore.

Step One

Step one reads as follows:

In `weather.dat` (bit.ly/ksbVPs) you'll find daily weather data for Morristown, N.J., for June 2002. Download this text file, then write a program to output the day number (column one) with the smallest temperature spread (the maximum temperature is the second column, the minimum the third column).

The `weather.dat` file looks like what's shown in **Figure 1** (except it stretches out for the full 30 days).

It's immediately clear that the file is not comma-separated, but positionally separated—the “MxT” (max temp) column always starts at the same place and the “MnT” (min temp) column starts at another fixed location. A glance at the file in Visual Studio reveals that each line is precisely 90 characters long; parsing this file to a line-by-line, string-by-string arrangement will be trivial, because we can split on newlines or 90-character lengths.

After that, however, things become less clear. Though the exercise asks us to output only the smallest temperature spread for the month by hardcoding position values and doing a `String.Subset` on each line, that misses some of our goals. So let's take things just

a step further and refine the problem to also require the ability to examine any data element for any day within the month.

Remember, the core of multiparadigmatic design is to identify the commonality/variability axis, and for this problem thus far, the commonality/variability axis is pretty unintuitive—though it’s easy to see that we need to parse a text file and examine the results. That data is essentially tabular in nature, and can be looked at in several different ways based on the paradigms we’ve investigated.

From a procedural perspective, the data structure is the definition of each line, and an object-oriented solution doesn’t give us much more. It’s fairly easy to simply take the procedural data structure and toss in the methods to parse the data on the class. Essentially, we’re back to “smart data,” which although not particularly object-ish, works. For the moment.

Nothing here suggests the practical use of meta-programming tactics yet, or of dynamic programming, except perhaps the data structure allowing some kind of lookup based on column name, so that `day1["MxT"]` would return “88.” The functional approach could be interesting, because parsing can be seen as a collection of functions, running one after another, taking input and returning strings (or other parsed data) and more functions to parse the rest of the file as results. Such a technique is known as parser combinators and is a discussion well beyond the scope of this particular article.

None of these approaches seem to help much; so far, the best solution seems to be a procedural approach, doing something like what’s shown in **Figure 2**.

Note that this code already presents a problem; when the parsing reaches day 9, an asterisk shows up in the MnT column, indicating that this is the “low” for the month, just as day 26 is the “high” for the month. This is solvable by stripping the “*” out of the string, but the point is made: The procedural axis focuses on establishing the data structure and operating on it—in this case, parsing it.

Also, note the use of `List<>` here; just because we’re using a procedural approach to parse the file doesn’t mean we can’t take advantage of useful classes elsewhere within the Base Class Library. This makes it trivial to figure out the smallest spread—a single LINQ-to-Objects query will give the needed result. (We should, of course, grab which day that spread occurs on, to be faithful to the problem, but that’s really just a matter of returning and `Max()`ing on something other than a raw float; consider that an exercise for the reader.)

Step Two

We could spend a lot of time projecting how we might write “reusable” code for this, but doing so would be like trying to predict the future; this code works, so let’s move on to the next step in the kata:

The file `football.dat` (bit.ly/lyNLYa) contains the results from the English Premier League for 2001/2002. The columns labeled “F” and “A” contain the total number of goals scored for and against each team in that season (so Arsenal scored 79 goals against opponents, and had 36 goals scored against them). Write a program to print the name of the team with the smallest difference in “for” and “against” goals.

More text parsing. Joy. The `football.dat` file is shown in **Figure 3**; it’s similar in many ways to `weather.dat` (**Figure 1**), yet different enough to merit some different parsing code.

The core of multiparadigmatic design is to identify the commonality/variability axis.

If we consider these two programs independently, it’s pretty easy to see that, individually, each is solved in much the same way—there’s not a lot of point in rehashing that exercise for football.

Step Three

The last step to the kata, however, leads us to the pay dirt of the exercise:

Take the two programs written previously and factor out as much common code as possible, leaving you with two smaller programs and some kind of shared functionality.

This is intriguing, and factors right in to commonality/variability analysis.

Commonality/Variability

Because we now have more than just one problem to examine—that is to say, a family of concerns—it becomes easier to tease out the common and the variable between the two problems. That analysis, starting with analysis of the text files, reveals the exercise essentially boils down to two steps: parsing the file into a list of row data, and then examining that data in terms of some kind of calculation or analysis.

Figure 1 The Weather.dat Text File

MMU June 2002																
Dy	MxT	MnT	AvT	HDDay	AvDP	1HrP	TPcpn	WxType	PDir	AvSp	Dir	MxS	SkyC	MxR	MnR	AvSLP
1	88	59	74		53.8		0.00	F	280	9.6	270	17	1.6	93	23	1004.5
2	79	63	71		46.5		0.00		330	8.7	340	23	3.3	70	28	1004.5
3	77	55	66		39.6		0.00		350	5.0	350	9	2.8	59	24	1016.8
4	77	59	68		51.1		0.00		110	9.1	130	12	8.6	62	40	1021.1
.	..															
28	84	68	76		65.6		0.00	RTFH	280	7.6	340	16	7.0	100	51	1011.0
29	88	66	77		59.7		0.00		040	5.4	020	9	5.3	84	33	1020.6
30	90	45	68		63.6		0.00	H	240	6.0	220	17	4.8	200	41	1022.7
mo	82.9	60.5	71.7	16	58.8		0.00			6.9			5.3			

Figure 2 Taking a Procedural Approach

```
namespace DataMunger
{
    public struct WeatherData
    {
        public int Day;
        public float MxT;
        public float MnT;
        // More columns go here
    }
    class Program
    {
        static void Main(string[] args)
        {
            TextReader reader = new StreamReader("weather.dat");

            // Read past first four lines
            reader.ReadLine();
            reader.ReadLine();
            reader.ReadLine();
            reader.ReadLine();

            // Start reading data
            List<WeatherData> weatherInfo = new List<WeatherData>();
```

```
while (reader.Peek() > 0)
{
    string line = reader.ReadLine();

    // Guard against "mo" summation line
    if (line.Substring(0, 4) == " mo")
        continue;

    WeatherData wd = new WeatherData();
    wd.Day = Int32.Parse(line.Substring(0, 4).Replace("*", " "));
    wd.MxT = Single.Parse(line.Substring(5, 6).Replace("*", " "));
    wd.MnT = Single.Parse(line.Substring(12, 6).Replace("*", " "));
    // More parsing goes here

    weatherInfo.Add(wd);
}

Console.WriteLine("Max spread: " +
    weatherInfo.Select((wd) => wd.MxT - wd.MnT).Max());
}
```

There are some issues to consider when parsing the text files:

- Both file formats have a “header” that needs to be ignored.
- Both file formats are positionally based.
- Both file formats have lines that will need to be ignored (the “mo” summation line in `weather.dat`, the “-----” visual marker in `football.dat`).
- `Weather.dat` has some empty columns; `football.dat` does not.
- Both file formats support principally numeric and string columns (except that `weather.dat` also includes “*” values, which may need to be captured somehow).

Calculating the results depends strongly on what the parsed data ends up like, so it seems reasonable to begin there. We have several approaches, based on each paradigm, from which we could start:

Procedural This axis focuses on capturing commonality in data structures. However, with two different file formats, that’s clearly where we want to capture variability, so procedural looks like it’s getting the boot. It might be possible to create some kind of data structure that captures both file formats, but it’s probably going to be awkward compared to the other paradigms.

The dynamic approach offers more flexibility at the cost of more complexity internally.

Object-Oriented The commonality between the two files suggests using a base abstract “TextParser” class to provide base parsing functionality, including the ability to skip lines. Variability comes with parsing each line, which means that subclasses must override some kind of “ParseLine” method to do the actual line-by-line parsing. How the parsed values are retrieved out of the TextParser subtype, however—in order to do the min/max comparison—could be tricky, because the types of the columns will also vary. Historically, the Microsoft .NET Framework solved this (with SQL datasets) by returning objects, which we could use if necessary. But that

introduces potential type-safety errors, because objects would need to be downcast to be useful, and that might be dangerous.

Meta Several different solutions present themselves along the meta-object/meta-programmatic range. The attributive approach suggests that the TextParser class could take a “Record” type, in which each described column has a start/length custom attribute describing how to parse the lines, like so:

```
public struct WeatherData
{
    [Parse(0, 4)]
    public int Day;
    [Parse(5, 6)]
    public float MxT;
    [Parse(12, 6)]
    public float MnT;
}
```

The TextParser could then be parameterized to take the Record type (TextParser<RecordT>) and use the custom attributes at run time to discover how to parse each line. This would then return a List of Records (List<RecordT>), from which we could do the aforementioned calculations.

Alternatively, generative programming suggests that a source format of some kind describes the text file, and the parser for each kind of file is generated based on that source format.

Dynamic Taking a dynamic approach is a bit strange, given it’s a newer approach for .NET, but here we might imagine a TextParser class taking a string that describes how to parse each file. It’d be almost like a tiny programming language in its own right, perhaps incorporating regular expressions, or (in our case) just using positional data, because that’s all that’s required for this problem:

```
string parseCommands =
    "Ignore; Ignore; Ignore; Ignore; Repeat(Day:0-4, MxT:5-6, MnT:12-6)";
TextParser parser = new TextParser(parseCommands);
List<string> MxTData = parser["MxT"];
List<string> MnTData = parser["MnT"];
```

This is clearly different from the meta-approach in that it lacks the type-safety the meta-approach could provide. However, a meta-approach requires compile-time changes if the file format changes, whereas a dynamic one, because it acts on name-based variability, could roll with the changes. The dynamic approach offers more flexibility at the cost of more complexity internally.

Figure 3 The Football.dat Text File

	Team	P	W	L	D	F	A	Pts
1.	Arsenal	38	26	9	3	79	- 36	87
2.	Liverpool	38	24	8	6	67	- 30	80
3.	Manchester_U	38	24	5	9	87	- 45	77
4.	Newcastle	38	21	8	9	74	- 52	71
5.	Leeds	38	18	12	8	53	- 37	66
6.	Chelsea	38	17	13	8	66	- 38	64
7.	West_Ham	38	15	8	15	48	- 57	53
8.	Aston_Villa	38	12	14	12	46	- 47	50
9.	Tottenham	38	14	8	16	49	- 53	50
10.	Blackburn	38	12	10	16	55	- 51	46
11.	Southampton	38	12	9	17	46	- 54	45
12.	Middlesbrough	38	12	9	17	35	- 47	45
13.	Fulham	38	10	24	14	36	- 44	44
14.	Charlton	38	10	14	14	38	- 49	44
15.	Everton	38	11	10	17	45	- 57	43
16.	Bolton	38	9	13	16	44	- 62	40
17.	Sunderland	38	10	10	18	29	- 51	40
18.	Ipswich	38	9	9	20	41	- 64	36
19.	Derby	38	8	6	24	33	- 63	30
20.	Leicester	38	5	13	20	30	- 64	28

The source format used by the generative meta-approach could be what's passed at run time, building the parser at run time instead of at source time.

Functional In some ways, the functional approach is the most curious, as it would suggest that the algorithm—the parsing—is where the variability lies, and therefore the TextParser class should take one or more functions that describe how to parse the text, whether line-by-line or column-by-column, or both. For example, two things the TextParser needs to know how to do are ignore non-parseable lines and break a line down into constituent parts. These could be established as function instances on TextParser itself, passed in either through a constructor or set via properties, as shown in **Figure 4**.

Figure 4 A Functional Approach

```
TextParser<WeatherData> parser = new TextParser<WeatherData>();
parser.LineParserVerifier =
    (line) =>
    {
        if ( (line.Trim().Length == 0) || // Empty line
            (line.Contains("MMU")) || // First header line
            (line.Contains("Dy")) || // Second header line
            (line.Contains("mo")))
            return false;
        else
            return true;
    };
parser.ColumnExtractor =
    (line) =>
    {
        WeatherData wd = new WeatherData();
        wd.Day = line.Substring(0, 4);
        wd.MxT = line.Substring(5, 6);
        wd.MnT = line.Substring(12, 6);
        return wd;
    };
List<WeatherData> results = parser.Parse("weather.dat");
```

TextParser uses the parameterized type solely to capture the WeatherData type for greater type-safety; it could be written to return generic System.Object if desired or if easier.

Wrapping Up

Clearly there's no real "one way" to approach the problem—as other requirements come into focus, we'd get a better sense of where the variabilities are, and thus a better sense of the commonalities that need to be captured. This, by the way, highlights the real power of refactoring: Because we can't predict the future, refactoring code means we can be wrong about our commonality/variability analysis and yet not have to "throw it all away and start over" when these errors come to light.

Multiparadigm design is not an easy subject to grasp, particularly not in one sitting. It took 10 articles to describe, and it's quite reasonable to expect that it will take much longer to incorporate into your own thinking. But doing so can lead to much more flexible and comfortable designs over time, and even offer solutions to thorny problems that leave other developers a little wild-eyed at how you managed to see that solution through the haze of the problem. Remember, in the end, a multiparadigmatic approach begins and ends with commonality and variability—keep that in mind, and lots of things will begin to sort themselves out.

Toward that end, I strongly suggest readers take the Data Munging kata and deliberately try to solve each of the two file-parse exercises using each of the five different paradigms separately, then look for ways to combine them to create powerful, reusable code. What happens when object-oriented is paired up with functional or dynamic? Does the solution get easier or more complicated? Then take the approach and throw a different file format at it—how does a CSV file format screw it up? More importantly, do this during some empty moments in between stories or project meetings; the experience gained from this will pay off in spades when you attempt to do this kind of design under the gun of a real production deadline.

As already mentioned but worth repeating: multiparadigm languages are here, in common use, and seem like they're here to stay. The various Visual Studio 2010 languages each exhibit some degree of each of these paradigms; C++, for example, has some parametric meta-programming facilities that aren't possible in managed code, owing to the way that the C++ compiler operates; and just recently (in the latest C++0x standard) it gained lambda expressions. Even the much-maligned ECMAScript/JavaScript/JScript language can do objects, procedures, meta-programming, and dynamic and functional paradigms; in fact, much of JQuery is built on these ideas. The sooner these ideas take root in your brain, the sooner you'll be able to code your way through the stickiest of situations.

Happy coding! ■

TED NEWARD is a principal with Neward & Associates, an independent firm specializing in enterprise .NET Framework and Java platform systems. He has written more than 100 articles, is a C# MVP, INETA speaker, and has authored or coauthored a dozen books, including "Professional F# 2.0" (Wrox, 2010). He consults and mentors regularly—reach him at ted@tedneward.com if you're interested in having him come work with your team, or read his blog at blogs.tedneward.com.

THANKS to the following technical expert for reviewing this article:

Anthony D. Green

Visual Studio® **LIVE!**

EXPERT SOLUTIONS FOR .NET DEVELOPERS

MICROSOFT HEADQUARTERS — REDMOND, WASHINGTON | OCTOBER 17-21

Sponsored by: **Microsoft®**

IN-DEPTH TRAINING ON THE MICROSOFT CAMPUS

Get ready for five action-packed days of **60+ sessions**, **9 tracks** and **5 workshops** led by industry heavyweights and Microsoft insiders — all taking place onsite at Microsoft Headquarters.

Reserve your spot today! Last year's event SOLD OUT.

SUPPORTED BY:

Microsoft®

 msdn

 Microsoft Visual Studio

Visual Studio
MAGAZINE

EVENT SPONSOR

Microsoft®

PLATINUM SPONSORS

 esri

VERSANT

WVS
WORLD OF VS

GOLD SPONSORS

Devexpress™
Download • Compare • Decide!

 CodeFluent
Entities

PRODUCED BY:

 1105 MEDIA®



5 DAYS OF TRAINING @ MICROSOFT HEADQUARTERS!

IF YOU'RE LOOKING FOR HARD-HITTING .NET DEVELOPER TRAINING, this event is for you! Join industry pros and Microsoft insiders for relevant, applicable and up-to-date education that will help you overcome development challenges.



TAKE YOUR CODE TO THE NEXT LEVEL WITH REAL-WORLD TRAINING ON:

- **Visual Studio 2010 / .NET**
- **Mobile Development**
- **Cloud Computing**
- **Web / HTML5**
- **Silverlight / WPF**
- **Data Management**
- **Developing Services**
- **Programming Practices**
- **LightSwitch**



REGISTER TODAY

VSLIVE.COM/REDMOND

Use priority code **SEPTAD**

VISUAL STUDIO LIVE! REDMOND

Silverlight/WPF	Developing Services	Visual Studio 2010/.NET 4	Cloud	Data Management	LightSwitch	Programming Practices	Web/HTML5	Mobile Dev
Visual Studio Live! Pre-Conference Workshops: Monday, October 17, 2011								
MWK1 Workshop: ALM in 2011: Visual Studio 2010 and the Next Big Release <i>Brian Randell</i>			MWK2 Workshop: Making Effective Use of Silverlight and WPF <i>Billy Hollis & Rockford Lhotka</i>			MWK3 Workshop: Programming with WCF in One Day <i>Miguel Castro</i>		
Visual Studio Live! Day 1: Tuesday, October 18, 2011								
Keynote: Visual Studio and a Glimpse of the Future, <i>Jason Zander</i> , Corporate Vice President of the Visual Studio Team in the Developer Division, Microsoft								
T1 Microsoft Session—Details TBA	T2 Intense Intro to Silverlight <i>Billy Hollis</i>	T3 AppFabric, Workflow and WCF—the Next-Generation Middleware <i>Ron Jacobs</i>	T4 If not IaaS, When Should I Use Windows: Windows Azure VM Role? <i>Eric D. Boyd</i>	T5 Best Kept Secrets in Visual Studio 2010 and .NET 4 <i>Deborah Kurata</i>				
T6 Microsoft Session—Details TBA	T7 XAML: Achieving Your Moment of Clarity <i>Miguel Castro</i>	T8 What's New in WCF 4 <i>Ido Flatow</i>	T9 What Is Microsoft Marketplace DataMarket? <i>Michael Stiefel</i>	T10 The LINQ Programming Model <i>Marcel de Vries</i>				
T11 Microsoft Session—Details TBA	T12 Fundamental Design Principles for UI Developers <i>Billy Hollis</i>	T13 Creating Scalable State Full Services Using WCF and WF <i>Marcel de Vries</i>	T14 Deciding Between Relational Databases and Tables in the Cloud <i>Michael Stiefel</i>	T15 NoSQL—Beyond the Key-Value Store <i>Robert Green</i>				
T16 HTML5 and Internet Explorer 9: Developer Overview <i>Ben Hoelting</i>	T17 Bind Anything to Anything in XAML <i>Rockford Lhotka</i>	T18 AppFabric Caching: How It Works and When You Should Use It <i>Jon Flanders</i>	T19 Microsoft Session—Details TBA	T20 How to Take WCF Data Services to the Next level <i>Rob Daigneau</i>				
Microsoft Ask the Experts & Welcome Reception								
Visual Studio Live! Day 2: Wednesday, October 19, 2011								
W1 HTML5 and Your Web Sites <i>Robert Boedigheimer</i>	W2 Handling Offline Data in Silverlight and Windows Phone 7 <i>John Papa</i>	W3 Microsoft Session—Details TBA	W4 Windows Azure Platform Overview <i>Vishwas Lele</i>	W5 Application Lifecycle Management and Visual Studio: What's Next <i>Brian Randel</i>				
W6 Styling Web Pages with CSS 3 <i>Robert Boedigheimer</i>	W7 Building Great Windows Applications with XAML and C# LINQ <i>Pete Brown</i>	W8 Building Native Mobile Apps with HTML5 & jQuery <i>Jon Flanders</i>	W9 Building Windows Azure Applications <i>Vishwas Lele</i>	W10 Visual Studio v.Next <i>Brian Randell</i>				
W11 The Best of jQuery <i>Robert Boedigheimer</i>	W12 What's New and Cool in Silverlight 5 <i>Pete Brown</i>	W13 Getting Started with Windows Phone 7 <i>Scott Golightly</i>	W14 Building Compute-Intensive Apps in Windows Azure <i>Vishwas Lele</i>	W15 Bringing Your Data and Maps Together with Esri Cloud Services <i>Arthur J. Haddad</i>				
W16 ASP.NET MVC, Razor and jQuery—the New Face of ASP.NET <i>Ido Flatow</i>	W17 Silverlight, WCF, RIA Services and Your Business Objects <i>Deborah Kurata</i>	W18 Windows Azure and Windows Phone—Creating Great Apps <i>Scott Golightly</i>	W19 Building and Running the Windows Azure Developer Portal <i>Chris Mullins</i>	W20 Microsoft Session—Details TBA				
Sponsor Reception / Wild Wednesday								
Visual Studio Live! Day 3: Thursday, October 20, 2011								
TH1 Creating a Data Driven Web Site Using WebMatrix and ASP.NET Razor <i>Rachel Appel</i>	TH2 Bringing the Silverlight PivotViewer to Your Applications <i>Tony Champion</i>	TH3 CSLA 4 for Windows Phone and Silverlight <i>Rockford Lhotka</i>	TH4 Microsoft Session—Details TBA	TH5 Design for Testability: Mocks, Stubs, Refactoring and UIs <i>Ben Day</i>				
TH6 How Orchard CMS Works <i>Rachel Appel</i>	TH7 MVVM in Practice aka “Code Behind”—Free WPF <i>Tiberiu Covaci</i>	TH8 Working with Data on Windows Phone 7 <i>Sergey Barskiy</i>	TH9 Microsoft Session—Details TBA	TH10 Team Foundation Server 2010 Builds: Understand, Configure and Customize <i>Ben Day</i>				
TH11 Busy Developer's Guide to (ECMA/Java)Script <i>Ted Neward</i>	TH12 Radically Advanced Templates for WPF and Silverlight <i>Billy Hollis</i>	TH13 Advanced Patterns with MVVM in Silverlight and Windows Phone 7 <i>John Papa</i>	TH14 Microsoft Session—Details TBA	TH15 Grasping The LightSwitch Paradigm (The Taming of the Tool) <i>Andrew Brust</i>				
TH16 Getting Started with ASP.NET MVC <i>Philip Japikse</i>	TH17 Using MEF to Develop Composable Applications <i>Ben Hoelting</i>	TH18 Windows Phone 7 Instrumentation—How to Learn from Your App <i>Tony Champion</i>	TH19 So Many Choices, So Little Time: Understanding Your .NET 4 Data Access Options <i>Lenni Lobel</i>	TH20 Static Analysis in .NET <i>Jason Bock</i>				
TH21 Test Driving ASP.NET MVC <i>Philip Japikse</i>	TH22 Patterns for Parallel Programming <i>Tiberiu Covaci</i>	TH23 Microsoft Session—Details TBA	TH24 Using Code First (Code Only) Approach with Entity Framework <i>Sergey Barskiy</i>	TH25 Modern .NET Development Practices and Principles <i>Jason Bock</i>				
Visual Studio Live! Post-Conference Workshops: Friday, October 21, 2011								
FWK1 Architectural Katas Workshop <i>Ted Neward</i>				FWK2 SQL Server Workshop for Developers <i>Andrew Brust & Leonard Lobel</i>				

→ INTRODUCING YOUR GURUS

Visual Studio Live! offers in-depth training for all levels of developers to help simplify the development process from design to deployment. Who better to learn from than Microsoft Team members and industry experts?

KEYNOTE



Jason Zander

Corporate Vice
President of the
Visual Studio Team
in the Developer
Division
Microsoft



Rachel Appel

Developer Evangelist
Microsoft



Andrew Brust

Founder and CEO
Blue Badge Insights



Billy Hollis

Next Version
Systems



Rockford Lhotka

Principal Technology
Evangelist
Magenic

DON'T JUST TAKE OUR WORD FOR IT

HERE'S WHAT PAST VISUAL STUDIO LIVE!
ATTENDEES HAD TO SAY:

"There were several sessions that will change the way I program."

*Monte Ellis, CTO, CIO,
KG Storagemart*

"Fun, informative conference! Loved learning about different technologies and talking to others in the industry."

*Peter C. Hoffman, ASP.NET
programmer, KBS*

"[Visual Studio Live!] introduced us to the latest and greatest Microsoft Technologies, tips and tricks, and real experiences to enable us to design, develop, and deliver best possible solutions to our clients. Thank you!"

*Sophia Mazurczyk, Dev. Team
manager, Tidgewell Associates, Inc.*



REGISTER TODAY

VSLIVE.COM/REDMOND

Use priority code **SEPTAD**



Touch for Text

My first store-bought computer was an Osborne 1, which was the first commercial computer designed to be small enough to fit under the seat of a plane. It accomplished this feat in part by featuring a tiny 5-inch monitor capable of displaying only 24 lines of 52 characters each.

My most recent computer purchase was a Windows Phone 7-based device, and of course it has an even smaller screen. The aspect ratio is different (3:5 rather than 4:3) and the Windows Phone 7 fonts have proportional spacing, but the text density is nearly identical: Silverlight programs written for a Windows Phone 7 device use a default font that displays approximately the same number of lines with the same number of characters as the Osborne 1 display!

I won't bore you with the myriad of differences between these two computers created 30 years apart, except to note that nobody ever considered reading a book on the Osborne 1. (Writing a book, yes, but reading a book, definitely not.) In contrast, reading books has become one of my favorite activities on my Windows Phone 7 device. I like reading books on my phone so much that I'm writing my own e-book-reading software so I can tailor the experience just the way I want it.

Touching the Words

Certainly much of the appeal of using the phone as an e-book reader is its ideal size: If the screen were any smaller, it would be hard to read and handle. Any larger, and it wouldn't fit in my pocket, and that's an important criterion.

I think the touch interface on the phone is a significant aspect of the appeal as well. The ease of flipping pages with a simple tap or a swipe of a finger seems to satisfy two requirements of a book: the tactile and the cerebral. The phone feels nothing like a real book, of course, but the responsiveness is so effortless that the device fades into the background so as to not interfere with the more cerebral experience of reading.

A touch interface is both a blessing and a curse for e-book readers. It's great for vague gestures that can be handled with leniency, such as dragging a scrolled page up or down, or flicking the pages of a paginated document forward or backward. The touch interface becomes awkward for activities that require more precision, such as text selection. The sheer difference in size between your fingers and the words on the screen makes text selection nearly impossible without some special assistance from the program. Yet, this

assistance must itself feel natural. If text selection turns into a complex process from the user's perspective, it might as well not be a program feature at all.

While it's easy to imagine an e-book reader without text selection, the feature offers too many benefits to ignore. An e-book reader can let the user look up a selected word or phrase in a dictionary, Wikipedia or a search engine. A selected sentence might be saved along with a reader's note such as "Great insight!" or "That doesn't follow at all." Or you might want to paste a selection from the book into an e-mail.

Though I'll be speaking about text selection in the context of an e-book reader, the concepts can be applied to any Windows Phone 7 program that displays text to the screen and allows the reader to interact with that text.

The Text-Selection Quandary

The first problem in text selection is hit-testing. When the user touches the screen, what word is underneath that finger?

Windows Phone 7 has three programming interfaces for touch: the low-level `Touch.FrameReported` event, the higher-level `Manipulation` events and (my favorite) the `Gesture` interface available with the Silverlight for Windows Phone Toolkit, which is downloadable from CodePlex. Each of these interfaces lets you determine the element at the point where a finger touches the screen. For text, this element is probably a `TextBlock`.

What you *cannot* do, however, is easily determine what part of the text displayed by the `TextBlock` the user is touching. If the `TextBlock` is displaying multiple words, you can't distinguish among these words without performing calculations using font metric information. And if you have the necessary font metric information—for example, based on the techniques I described in last month's installment of this column—you're better off using that information to separate text into multiple words where each `TextBlock` is devoted to displaying a single word.

If each word is a `TextBlock`, it's easy to determine which word is at the point on the screen corresponding to the user's finger, and then to track the user's finger in selecting multiple consecutive words. However, the user might be more uncertain than the program! Fingers are not transparent, and fingertips often dwarf the tiny words on the screen.

To accurately select individual words on the screen, the user needs to zoom in on the screen before selecting the text. With multi-touch interfaces, the standard way to expand an element on the screen is a two-finger pinch operation. However, pinch is only part of it. Once the screen is magnified, the user should be able to

Code download available at code.msdn.microsoft.com/mag201109UIFrontiers.

Figure 1 The Content Area of BookViewer.xaml

```
<Grid x:Name="LayoutRoot">
  <toolkit:GestureService.GestureRecognizer>
    <toolkit:GestureRecognizer GestureBegin="OnGestureRecognizerGestureBegin"
      GestureCompleted="OnGestureRecognizerGestureCompleted"
      Tap="OnGestureRecognizerTap"
      Hold="OnGestureRecognizerHold"
      Flick="OnGestureRecognizerFlick"
      DragStarted="OnGestureRecognizerDragStarted"
      DragDelta="OnGestureRecognizerDragDelta"
      DragCompleted="OnGestureRecognizerDragCompleted"
      PinchStarted="OnGestureRecognizerPinchStarted"
      PinchDelta="OnGestureRecognizerPinchDelta"
      PinchCompleted="OnGestureRecognizerPinchCompleted" />
  </toolkit:GestureService.GestureRecognizer>

  <Grid Name="manipulationGrid" CacheMode="BitmapCache">
    <Border Name="pageContainer0" Style="{StaticResource pageContainerStyle}">
      <Border Name="pageHost0" Style="{StaticResource pageHostStyle}" />
    </Border>

    <Border Name="pageContainer1" Style="{StaticResource pageContainerStyle}">
      <Border Name="pageHost1" Style="{StaticResource pageHostStyle}" />
    </Border>

    <Border Name="pageContainer2" Style="{StaticResource pageContainerStyle}">
      <Border Name="pageHost2" Style="{StaticResource pageHostStyle}" />
    </Border>

    <Grid.RenderTransform>
      <TransformGroup x:Name="transformGroup">
        <MatrixTransform x:Name="matrixTransform" />
        <ScaleTransform x:Name="scaleTransform" />
        <TranslateTransform x:Name="translateTransform" />
      </TransformGroup>
    </Grid.RenderTransform>
  </Grid>

  <!-- Scratch pad transforms -->
  <Grid Width="0" Height="0">
    <Grid.RenderTransform>
      <TransformGroup x:Name="calcTransformGroup">
        <MatrixTransform x:Name="calcMatrixTransform" />
        <ScaleTransform x:Name="calcScaleTransform" />
        <TranslateTransform x:Name="calcTranslateTransform" />
      </TransformGroup>
    </Grid.RenderTransform>
  </Grid>
</Grid>
```

use a single finger to shift the page around relative to the viewport of the screen. This is sometimes known as a “pan” operation, from the use of the word in motion picture camerawork.

In short, implementing text selection means dealing with touch events that are interpreted differently for different modes of operation. A drag operation—touching the screen and moving the finger—normally has the effect of shifting the page forward or back. But a drag operation is also required to extend a text selection beyond one word, which means that something is needed to put the program into a text-selection mode.

To signal when a selection begins, an excellent choice is the gesture called “hold.” This occurs when the user presses a finger to the screen and holds it still for about a second. Any drag operation that follows the hold gesture is interpreted as extending the selection.

However, if the user has used a pinch operation to expand the screen prior to making the selection, then a regular drag—that is, a drag that’s not preceded by a hold—must be interpreted as a panning operation rather than a page transition.

In summary, drag operations can be interpreted in three different ways depending on the current mode.

Implementing Gesture Modes

If you’ve been following the past few installments of this column, you know I’ve been progressively building a Windows Phone 7 e-book reader by isolating various features and exploring them. The downloadable program for this article is called BleakHouseReader. Like the other programs, it’s restricted to one book, this time Charles Dickens’ 1853 novel “Bleak House,” which is not the total downer the title would seem to indicate. The book is a public domain plain text file downloaded from the Project Gutenberg Web site.

The previous program (PhineasReader) showed the performance improvements in pagination when you switch from using a TextBlock for an entire paragraph (or as much of a paragraph as can fit on one page) to using a separate TextBlock element for each word on the page. Besides providing faster layout, that switch was also the necessary first step to implementing text selection.

The BookViewer control in BleakHouseReader installs handlers for all but one of the touch gestures, as shown in the core of the BookViewer.xaml file in **Figure 1**. The only gesture the control ignores is DoubleTap. As usual, the three nested Border elements are used to host the current page, previous page and next page. Additions to BookViewer.xaml for this program are the two blocks of transforms used for pinch and pan operations. The second block of transforms lets the program do some transform calculations without explicitly performing matrix multiplication.

The codebehind file for BookViewer.xaml is BookViewer.xaml.cs, but to prevent that file from becoming too unwieldy, I created a second codebehind file named BookViewer.Gestures.cs specifically for the 11 handlers for the 11 touch gestures.

Previously I used a couple of Boolean fields to help interpret complex combinations of gestures. For this program I switched to enumerations. **Figure 2** shows the ViewerTouchMode and ViewerDisplayMode enumerations. These help the gesture event handlers keep what’s happening straight and avoid collisions.

For example, a page transition begins with a tap, drag or flick gesture by the user, but ends with an animation. If any gesture begins when the current touch mode is ViewerTouchMode.Animating, the entire gesture is ignored. The handler for the hold gesture checks if a TextBlock is under the finger. If so, the current touch mode becomes ViewerTouchMode.Selecting until the finger lifts from the screen. When the touch mode is Selecting, a drag gesture extends the selection.

Figure 2 The ViewerTouchMode and ViewerDisplayMode Enumerations

```
public enum ViewerTouchMode
{
    Reading = 0,
    Dragging,
    Selecting,
    Pinching,
    Panning,
    Animating,
}

public enum ViewerDisplayMode
{
    Normal = 0,
    Zoomed
}
```


Figure 3 The ClampPinchAndPanTransforms Method

```
void ClampPinchAndPanTransforms(
    double scale, double translateX, double translateY)
{
    // This is the matrix transform from previous operations.
    calcMatrixTransform.Matrix = matrixTransform.Matrix;

    // Calculate scaling factor so it's always 1 or greater.
    double totalScale = scale * matrixTransform.Matrix.M11;
    totalScale = Math.Max(1, totalScale);
    scale = totalScale / matrixTransform.Matrix.M11;

    // Set up properties for new scale matrix.
    calcScaleTransform.CenterX = scaleTransform.CenterX;
    calcScaleTransform.CenterY = scaleTransform.CenterY;
    calcScaleTransform.ScaleX = scale;
    calcScaleTransform.ScaleY = scale;

    // Set up properties for new translation matrix.
    calcTranslateTransform.X = translateX;
    calcTranslateTransform.Y = translateY;

    // Obtain the total matrix from the transform group.
    Matrix totalMatrix = calcTransformGroup.Value;

    // Restrict translation to original area of transformed element.
    double totalTranslationX = totalMatrix.OffsetX;
    double totalTranslationY = totalMatrix.OffsetY;

    double clampedTranslationX =
        Math.Min(0,
            Math.Max((1 - totalMatrix.M11) * manipulationGrid.ActualWidth,
                totalTranslationX));

    double clampedTranslationY =
        Math.Min(0,
            Math.Max((1 - totalMatrix.M22) * manipulationGrid.ActualHeight,
                totalTranslationY));

    // Adjust translation factors.
    translateX += clampedTranslationX - totalTranslationX;
    translateY += clampedTranslationY - totalTranslationY;

    // Set transforms.
    scaleTransform.ScaleX = scale;
    scaleTransform.ScaleY = scale;
    translateTransform.X = translateX;
    translateTransform.Y = translateY;
}
```

If no fingers are touching the screen, the current touch mode is either `ViewerTouchMode.Reading` or `ViewerTouchMode.Animating`, and when the animation ends, the touch mode will switch to `ViewerTouchMode.Reading`. Aside from the `Animating` value, the touch mode only pertains to gestures currently in progress.

However, the user can pinch the screen to make it larger, and remove all fingers from the screen, and the screen should remain zoomed. Then the user can perform another pinch operation to make the screen still larger or smaller, and combine that with a drag operation to move it around. This is why the `ViewerDisplayMode` enumeration is required. For example, if the user drags a finger on the screen, that becomes a page transition if the current display mode is `ViewerDisplayMode.Normal`, but becomes a pan operation for `ViewerDisplayMode.Zoomed`.

Clamping Pinch and Pan

The current display mode is switched from `ViewerDisplayMode.Normal` to `ViewerDisplayMode.Zoomed` if the user touches the screen with two fingers and stretches his fingers apart to make the display larger. The display mode remains `ViewerDisplayMode.Zoomed` until the user taps the screen. The `BookViewer` control responds to the tap by animating the screen back to normal size and switching to `ViewerDisplayMode.Normal`. Alternatively, if the user pinches the screen and contracts it so the scaling factor drops below 1.05, the screen is automatically restored to normal size.

When the display mode is zoomed, any additional pinching and panning operations are compounded with those from previous operations. This is the reason for the three grouped transforms in **Figure 1**. During a single pan operation, the `TranslateTransform` properties are altered to move the `Grid` relative to the screen. During a single pinch operation, both the `ScaleTransform` and

`TranslateTransform` get involved. After the gesture has completed, the `TransformGroup` indicates the total transform. A short method named `ConsolidateTransform` in `BookViewer.xaml.cs` transfers this total transform into the `MatrixTransform`, and then sets the `TranslateTransform` and `ScaleTransform` back to default values in preparation for the next gesture operation. In this way, the `MatrixTransform` accumulates the effects of all the pinch and pan operations.

I also found it necessary to limit the extent of the pinch and pan operations. For example, it only makes sense for a pinch operation to result in a page-scaling factor greater than one. There's no reason to let the page shrink to less than its normal size. Similarly, it makes no sense to let the page be panned so we can see "underneath" the page. The left edge of the page shouldn't appear to the right of the left edge of the screen, and the equivalent for the other three sides.

These restrictions are applied in a method in `BookViewer.xaml.cs`

named `ClampPinchAndPanTransforms`, shown in **Figure 3**. This method was definitely one of the most difficult parts of this program. It uses the other three transforms in `BookViewer.xaml` for performing "scratchpad" matrix-transform calculations.

Left- and right-flick gestures normally initiate page transitions and are implemented with animations to go to the next or previous page. For this version, I decided that flick gestures going up or down should insert a bookmark at that page, but those bookmarks haven't been implemented yet.

But what about flick gestures when the current display mode is `ViewerDisplayMode.Zoomed`? It seems as if flick gestures should move the page relative to the viewport. I decided to simply move the zoomed page to the extreme left, top, right or bottom of the viewport, depending on the angle of the flick gesture. This is one part of the implementation that I'm not entirely happy about. The movement should really be animated to seem as if the page is moving as a result of momentum and then slowing down.

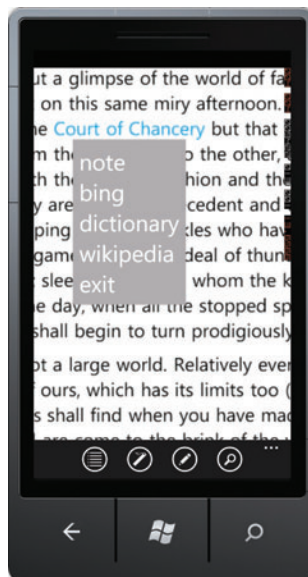


Figure 4 A Zoomed Page with a Text-Selection Menu

When the user presses a finger to the screen over a word for a second, moves that finger to extend the selection and then lifts the finger, the BookViewer control fires a `TextSelected` method. `MainPage` handles this event by displaying a little menu on top of the page, as shown in **Figure 4**.

The first item is not yet implemented. This feature will allow the user to type in a little note about the selected passage. These notes will be saved with document settings. The last item simply dismisses the menu. The “bing” item uses the Windows Phone 7 `SearchTask` class to invoke the phone’s standard Web search application. The other two items use the `WebBrowserTask` to invoke Internet Explorer with a URL that includes a query string with the text selection.

For the dictionary, I originally wanted to use the Bing dictionary, but in practice it seemed rather erratic. Sometimes it didn’t seem like a dictionary at all. After some further exploration, I ended up with the Google dictionary, which provides extensive results that seemed more suitable for my needs.

Second Thoughts

I mentioned that the “notes” option on the text-selection menu is unimplemented, as well as the feature to insert bookmarks with flick gestures. That’s not all that’s not yet implemented! In the image in **Figure 4**, the first button on the application bar brings up the chapter list. That works. The other three buttons will eventually bring up a list of all bookmarks, a list of all notes and a dialog to search for text in the book—but not quite yet.

As I’ve been reading “Bleak House,” I’ve enjoyed looking up words or phrases in Bing, Wikipedia and the Google dictionary. It’s fairly simple to expand the page, select the word or phrase and then tap the menu item.

However, I can tell already that this is *not* the best approach for selecting text for the notes feature. Text selected for making a note is almost always at least a sentence in length. But once the page is expanded, part of that sentence will probably be off the screen, and there’s currently no way to pan the screen while making a selection.

I’m wondering now if text selection for the notes feature should be a little different. I’m wondering if a different text-selection scheme could automatically kick in if the page is not zoomed. I’m wondering if a hold gesture on an unzoomed page should select an entire sentence, and then dragging should extend the selection to other whole sentences.

That’s the great thing about software: Nothing is ever fully committed. There are always opportunities for enhancements and improvements. ■

CHARLES PETZOLD is a longtime contributing editor to MSDN Magazine. His recent book, “Programming Windows Phone 7” (Microsoft Press, 2010), is available as a free download at bit.ly/cpebookpdf.

THANKS to the following technical expert for reviewing this article:
Chipalo Street

we are Countersoft you are Control



GEMINI
Project Platform



**BOOK A DEMO
FREE TRIAL**
www.geminipatform.com

**The most versatile project management
platform for software development and
testing teams around the world.**

*Allows teams to work the way they want to work with more
functionality and easy customization for optimum team performance.*



ATLAS
Product Support Optimized



DOWNLOAD NOW
and lead from the front
www.atlasanswer.com

**Because in product support you
should only answer any question once.
Knowledge to the people!**

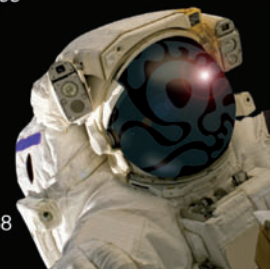
*A new generation of community support and self-help software.
Integrates Q&A, Knowledge Base, FAQs, Documents, Videos
and Podcasts and a simple, feature-rich User Interface.*



COUNTERSOFT

ENABLING
COLLECTIVE
CAPABILITY

Europe/Asia: +44 (0)1753 824000 // US/Canada: 800.927.5568
sales@countersoft.com www.countersoft.com





Development /s Design

Too many developers and managers think that user experience (UX) design is selecting colors and fonts and button radii. Nothing could be further from the truth. The rounded window corners and cutesy animations are the last and least important piece of the UX. My fellow Software Legend Billy Hollis calls that stuff “decoration, not design.”

Your entire program *is* the UX: its workflows, its feature sets, its required inputs. The UX isn't something that you throw over the fence to those artsy guys in berets who will somehow decorate your sow's ear of a program into a silk purse.

The UX battle is won or lost long before the program reaches the decorators. Think of your program as a piece of furniture—say, a table. The decoration is the surface finish on that table. Certainly tables should be finished well rather than poorly. But if the developer builds the table from the wrong material, one that doesn't satisfy the user's needs, even the best finish in the world can't help. If your user wants a decorative table for a nature-inspired living room, choosing wood will probably make him happy. On the other hand, if your customer needs a working table for a cafeteria kitchen that undergoes daily sanitizing, metal would be far better. And backing up a step, does the user really need a table, or would a chair solve his problem better?

Here's an example of fundamental development decisions making or breaking a UX. I use it in my Why Software Sucks keynote talk.

I was teaching in Sweden some time ago and browsed to Google. Its servers detected the country and automatically replied in Swedish (**Figure 1**). That's correct for most users most of the time, and requires just one click (lower right) to fix it permanently (persistent cookie) if it's ever wrong. On the other hand, UPS.com requires the user to select his country before he can do anything at all. That takes 30 clicks to do if you're in Sweden, and you have to explicitly tell the site to remember you or it'll make you do it again next time. That's no way to treat a customer whose money you want.

The UPS architects condemned their site to suck when they chose not to implement automatic detection of a user's country. (I've done it; it's not difficult. Simple static table lookup, update the tables once per day. Easy.) They showed contempt for the customers who pay their salaries, deliberately choosing to waste users' time doing what a computer could have and should have done for them. No decorator in the world can correct the results of this malpractice.

Google, on the other hand, understands that its aggregate users' time costs billions of dollars per hour, as I've written previously. Its automatic country detection, automatic topic suggestions and automatic fetching of search results all combine to answer the user's

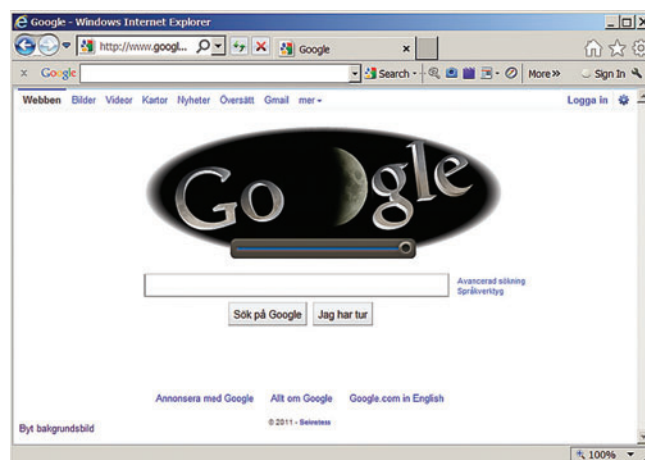


Figure 1 Automatic country detection is one reason why Google does it right.

question; quickly, correctly and with minimum possible effort. The world's worst decorator would have a hard time wrecking this site.

Google occasionally adds a topical decoration to its logo for a touch of whimsy. I recently saw a working guitar, a Charlie Chaplin spoof video and the lunar eclipse in **Figure 1**. But Google never lets this decoration obstruct its functionality, and never short-changes its functionality budget to fund it. Its official name, the “Google Doodle,” acknowledges its non-essential nature.

Clients sometimes ask me to critique their UXs just before they ship. That's way too late to change anything. The architecture is set, the budgets spent, the attitudes hardened. When I see the corners that my clients have backed themselves into, I want to tear my hair out. Like a cancer surgeon, I want to scream: “Why didn't you come to me a year ago when I *could* have helped you?” Like the surgeon, I provide whatever palliative care I can at that point. But as I do that, I silently swear to preach the gospel of early intervention, to spare others the pain I can't help this guy with.

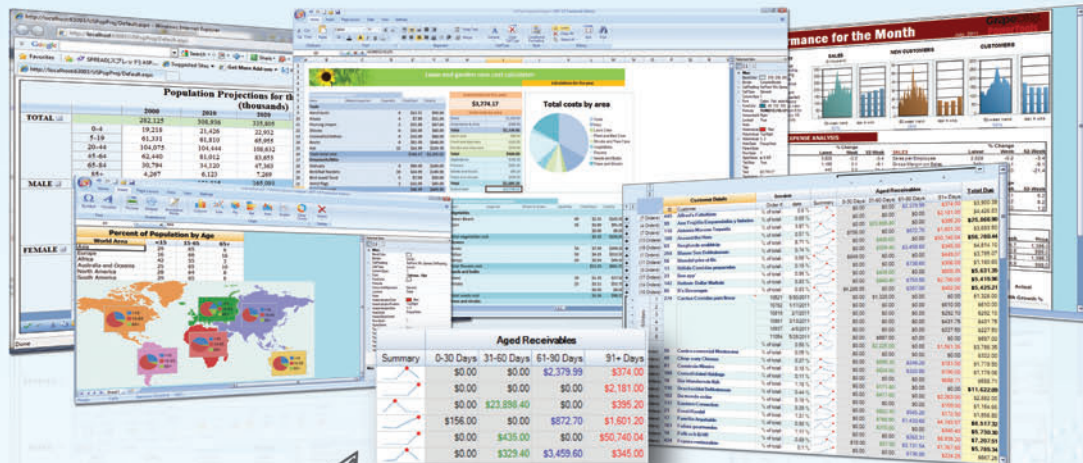
The UX starts at the beginning of the development project, not the end. Thank you for reading that. Now go do it. Call me sooner, not later. ■

DAVID S. PLATT teaches Programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including “Why Software Sucks” (Addison-Wesley Professional, 2006) and “Introducing Microsoft .NET” (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.

Think your grid is good enough?

*Excel documents out of control? Need complete control of your data?
Need to visualize and analyze your data? Need Excel-compatible function library?*

New Version!



Now with Sparklines!

Think again.

Smarter Components for Smarter Developers

Spread.NET allows you to embed the deep functionality and user experience found in Microsoft® Excel® into all of your .NET and ASP.NET applications, without requiring Microsoft Excel. Royalty-free. Run-time free.

Russ Fustino
Senior Developer Evangelist
Host, RUSS CAM

Celebrating 20 years of Excellence!



Download your FREE Trial Today!
www.GCPowerTools.com/SpreadNET
GvTv.GCPowerTools.com





The Dashboard Framework for Developers like you



V2.5 Now Available

Dundas Dashboard was built with developers and IT staff in mind. Whether it's our open API, simple web integration or powerful scripting capabilities, technologists have all the tools and options they need for getting their dashboard projects up and running quickly and easily.



Powered by
Microsoft Silverlight



www.dundas.com
(416) 467-5100 • (800) 463-1492

Silverlight is a trademark of Microsoft Corporation in the United States and/or other countries.